KERJA PRAKTIK - IF184801

Rancang Bangun Aplikasi Marketplace VEGGO Menggunakan Kerangka Kerja Laravel

DPTSI ITS
Gedung Pusat Riset Center Lantai 4, Kampus ITS Sukolilo, Surabaya
Periode: 01 November 2020 – 30 Desember 2020

Oleh:

I Gede Agung Krisna Pamungkas	05111740000135
Zaky Thariq H	05111740000140

Pembimbing Jurusan
Hadziq Fabroyir, S.Kom., Ph.D.

Pembimbing Lapangan
Rizky Januar Akbar, S.Kom., M.Eng.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK – IF184801

Rancang Bangun Aplikasi Marketplace VEGGO Menggunakan Kerangka Kerja Laravel

DPTSI ITS
Gedung Pusat Riset Center Lantai 4, Kampus ITS Sukolilo, Surabaya
Periode: 01 November 2020 – 30 Desember 2020

Oleh:

I Gede Agung Krisna Pamungkas	05111740000135
Zaky Thariq H	05111740000140

Pembimbing Jurusan
Hadziq Fabroyir, S.Kom., Ph.D.

Pembimbing Lapangan
Rizky Januar Akbar, S.Kom., M.Eng.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Rancang Bangun Aplikasi Sistem Marketplace VEGGO Menggunakan Kerangka Kerja Laravel

Oleh:

I Gede Agung Krisna P	05111740000135
Zaky Thariq H.	05111740000140

Disetujui oleh Pembimbing Kerja Praktik:

1. Hadziq Fabroyir, S.Kom., Ph.D.
   NIP 198602272019031006
   ..... (Pembimbing Departemen)

2. Rizky Januar Akbar, S.Kom., M.Eng.
   ..... (Pembimbing Lapangan)

SURABAYA
Desember, 2020

[Halaman ini sengaja dikosongkan]

Rancang Bangun Aplikasi Marketplace VEGGO Menggunakan Kerangka Kerja Laravel

Nama Mahasiswa	: I Gede Agung Krisna P
NRP	: 05111740000135
Nama Mahasiswa	: Zaky Thariq H.
NRP	: 05111740000140
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Hadziq Fabroyir, S.Kom., Ph.D.
Pembimbing Lapangan	: Rizky Januar Akbar, S.Kom., M.Eng

ABSTRAK

Direktorat Pengembangan Teknologi dan Sistem Informasi (DPTSI) bertugas untuk menyediakan dan mengelola layanan Teknologi Informasi di lingkungan ITS. Terkait peran, DPTSI berperan untuk mendukung aktivitas akademik, penelitian dan pengabdian masyarakat, serta manajerial di lingkungan ITS dalam rangka membantu ITS mencapai visi misinya.

Salah satu project yang sedang dikerjakan ketika kami melakukan Kerja Praktik adalah VEGGO, yaitu sistem marketplace penjualan kebutuhan pangan terutama sayur organik. Pengguna utama aplikasi yang kami kembangkan adalah pembeli yang melakukan pembelian secara tradisional yaitu melalui via Whatsapp. Selain itu aplikasi ini diperuntukkan untuk pihak VEGGO dalam rangka meningkatkan efektivitas pengelolaan transaksi.

Sistem ini dibangun menggunakan *framework* Laravel dengan menggunakan *database* MySQL serta beberapa *library* Javascript.

Kata Kunci: Website, VEGGO, Laravel

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik yang berjudul: Rancang Bangun Aplikasi Sistem Marketplace VEGGO Menggunakan Kerangka Kerja Laravel

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Bapak Hadziq Fabroyir, S.Kom., Ph.D. selaku dosen pembimbing kerja praktik selama kerja praktik berlangsung.
3. Bapak Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D selaku koordinator kerja praktik.
4. Bapak Rizky Januar Akbar, S.Kom., M.Eng selaku pembimbing lapangan selama kerja praktik berlangsung.

Surabaya, 30 Desember 2020

I Gede Agung K P
Zaky Thariq H.

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Saat ini dunia telah masuk ke dalam era digital dimana semua layanan dapat disajikan secara *online* dan berbagai teknologi informasi dapat dimanfaatkan untuk mempermudah manusia dalam memenuhikebutuhannya. Bila dibandingkan dengan masa lalu, ketika kita membaca koran atau majalah dengan membeli fisik koran atau majalah, kini, untuk membaca koran, kita dapat membuka website atau aplikasi penyedia koran untuk membacanya. *Website* atau aplikasi tidak hanya terbatas untuk membaca koran. Dengan website atau aplikasi manusia seakan-akan dapat melakukan segala hal dalam satu tempat. Banyak sekali lingkup fungsi dari website atau aplikasi teknologi informasi, sebagaimana contohnya *marketplace*, *e-banking*, pembelajaran, dan lain lain.

Pada kerja praktik (KP) ini, kami mencoba memanfaatkan teknologi informasi untuk meningkatkan dayaguna VEGGO. Sejauh ini, VEGGO melakukan bisnisnya secara manual dengan menggunakan aplikasi Whatsapp, mulai dari proses pemasaran hingga transaksi. Hal tersebut akan sangat merepotkan jika dikelola oleh satu orang. Maka dari itu dibutuhkan suatu sistem untuk mengelola proses bisnisnya dari proses pemasaran hingga transaksinya. Kesempatan KP ini kami dapatkan dari DPTSI. Kami diminta untuk menyelesaikan implementasi sistem proses bisnis VEGGO tersebut.

1.2. Tujuan

Tujuan kerja praktik ini adalah untuk menunaikan dua satuan kredit semester di perkuliahann kami dan untuk membantu pihak VEGGO dalam menyelesaikan permasalahan proses bisnis yang masih dilakukan secara manual dalam bentuk website.

1.3. Manfaat

Manfaat yang diperoleh dengan adanya *website marketplace* VEGGO antara lain adalah mempermudah proses pemasaran barang yang akan dijual, mempermudah proses transaksi, dan mempermudah merekap transaksi yang telah dilakukan. Jadi tidak perlu pergi bertransaksi melalui aplikasi Whatsapp. Cukup mengakses lewat website.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana analisis dan perancangan sistem *marketplace* VEGGO untuk para penggunanya: Pembeli, Penjual, Petani, Reseller?
2. Bagaimana implementasi sistem *marketpace* VEGGO untuk para penggunanya: Pembeli, Penjual, Petani, Reseller?

1.5. Lokasi dan Waktu Kerja Praktik

Lokasi kerja praktik berada di DPTSI dengan alamat Gedung Pusat Riset Lantai 4, Kampus ITS Sukolilo, Surabaya.

Adapun kerja praktik dimulai pada tanggal 1 November 2020 hingga 30 Desember 2020.

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi:

1.6.1 Perumusan Masalah

Untuk mengetahui domain dan fungsionalitas, kami mengikuti rapat bersama pihak VEGGO. Pihak VEGGO memiliki permintaan untuk pembuatan website marketplace dengan empat antarmuka. Pada saat rapat kami dijelaskan bagaimana konsep dan proses pemasaran hingga transaksi. Setelah dijelaskan, pemimpin tim *developer* merumuskan fitur-fitur apa saja yang akan diterapkan pada *website* yang akan dibuat.

1.6.2 Studi Literatur

Setelah mendapat gambaran bagaimana sistem tersebut berjalan, kami diberitahu tinjauan apa saja yang akan diimplementasikan untuk membuat *website* beroperasi. Tinjauan yang dipakai meliputi Laravel, MySQL, dan lain-lain.

1.6.3 Analisis dan Perancangan Sistem

Setelah tinjauan yang dipakai telah diberitahu, untuk merancang sistem yang baik perlu adanya sebuah desain arsitektur sistem. Pada *website*

ini tim *developer* setuju menggunakan arsitektur desain MVC (Model - View - Controller).

1.6.4 Implementasi Sistem

Implementasi merupakan realisasi dari tahap perancangan. Pada tahap ini kami membuat *website* yang sudah dirancang oleh pengembang yang sebelumnya.

1.6.5 Pengujian dan Evaluasi

Setelah *website* yang telah direncanakan telah jadi, perlu adanya evaluasi untuk menguji apakah *website* sesuai dengan harapan klien. Jika masih belum sesuai atau perlu menambah fitur, rapat akan dilakukan lagi untuk memfloorkan fitur-fitur apa saja yang perlu diperbaiki atau ditambah.

1.6.6 Kesimpulan dan Saran

Pengujian yang dilakukan ini telah memenuhi syarat yang diinginkan, dan berjalan dengan baik dan lancar.

1.7. Sistematika Laporan

1.7.1 Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2 Bab II Profil Perusahaan

Bab ini berisi gambaran umum Perusahaan DPTSI mulai dari profil, lokasi perusahaan.

1.7.3 Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4 Bab IV Analisis dan Perancangan Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

1.7.5 Bab V Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi aplikasi.

1.7.6 Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7 Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL PERUSAHAAN

2.1. Profil DPTSI

Direktorat Pengembangan Teknologi dan Sistem Informasi (DPTSI) bertugas untuk menyediakan dan mengelola layanan Teknologi Informasi di lingkungan ITS. Terkait peran, DPTSI berperan untuk mendukung aktivitas akademik, penelitian dan pengabdian masyarakat, serta manajerial di lingkungan ITS dalam rangka membantu ITS mencapai visi misinya. Visi dari DPTSI adalah Membangun Budaya dan Transformasi Digital untuk Peningkatan Kualitas Kinerja dan Layanan Berbasis TIK dalam rangka mewujudkan ITS sebagai *"World Class Research and Innovative University"*. Dan misi dari DPTSI adalah sebagai berikut:

1. Peningkatan Kapasitas Jaringan
2. Pengembangan & Pemeliharaan Sistem Informasi Terintegrasi menuju ITS in One Hand
3. Peningkatan Efisiensi dan Efektifitas Pelaporan dengan ITS Satu Data
4. Peningkatan Layanan Prima berbasis Digital

Adapun tujuan dari DPTSI sebagai berikut:

- Meningkatkan SDM yang profesional.
- Meningkatkan aksesibilitas informasi.
- Meningkatkan proses efisiensi.
- Menyediakan pelayanan dan support.
- Mengikuti dan mengembangkan teknologi informasi.

2.2. Lokasi

Gedung Pusat Riset Lantai 4, Kampus ITS Sukolilo, Surabaya.

BAB III
TINJAUAN PUSTAKA

3.1. Pemrograman Web

Web atau World Wide Web adalah ruang informasi yang berisi dokumen dan resource web lainnya yang dapat diidentifikasi melalui sebuah URL (Uniform Resource Locators, contohnya www.google.com) dan diakses ketika terkoneksi dengan internet. Halaman penyedia dokumen di dalam web dapat disebut sebagai *website* yang dapat terkoneksi satu dengan lainnya (hyperlink).

Pemrograman web adalah proses pembuatan halaman web agar bisa diakses oleh semua orang. Dalam pembuatan *website*, diperlukan sebuah standar pada *website* agar semua orang dapat membaca informasi dalam keadaan yang berbeda. Standar tersebut adalah HTML (Hypertext Markup Language). Jadi pemrograman web memiliki tugas untuk menciptakan suatu halaman sesuai standar HTML agar semua orang memiliki akses pada informasi di dalam halaman tersebut.

3.2. HTML

Bahasa standar internasional yang digunakan untuk membuat halaman web. HTML menggambarkan struktur dan isi semantik dari sebuah dokumen. HTML biasanya digabungkan dengan CSS dan Javascript. CSS untuk memperindah tampilan dan Javascript untuk *client-side scripting language*.

3.3. Javascript

Javascript adalah sebuah bahasa tingkat tinggi yang dinamis. Javascript memiliki banyak sekali fungsionalitas seperti *web application, backend, desktop application, internet of things* (IoT), dan lain-lain. Pada buku kerja praktik ini Javascript digunakan untuk *clientside scripting language* yang tertanam pada HTML sebuah *website*. Javascript juga memiliki banyak *library* yang dapat digunakan contohnya Nodejs, Axiosjs, Bluebirdjs, Vuejs, Angularjs, Reactjs, Animatejs, dan lain-lain.

3.4. Laravel

Laravel adalah sebuah *web application network* yang bersifat *open-source* yang digunakan untuk membangun aplikasi php dinamis. Laravel menjadi sebuah *framework* PHP dengan model MVC (Model, View, Controller) untuk membangun *website* dinamis dengan menggunakan PHP yang dapat mempercepat pengembang untuk membuat sebuah aplikasi web. Selain ringan dan cepat, Laravel juga memiliki dokumentasi yang super lengkap disertai dengan contoh implementasi kodenya. Dokumentasi yang lengkap inilah yang menjadi salah satu alasan kuat mengapa banyak orang memilih Laravel sebagai *framework* pilihannya.

3.5. MySQL

Merupakan salah satu sistem manajemen relasional basis data SQL yang bersifat *open source.* MySQL digunakan untuk menyimpan data - data yang dapat saling berelasi dengan data yang lain. Untuk melakukan operasi CRUD (Create, Read, Update, Delete) memerlukan *query*, lalu *query* tersebut dikirim ke dalam

database MySQL. Pada saat *query* tersebut sampai pada *database*, *database* akan mengolah *query* tersebut dan diterapkan sesuai perintah *query*.

3.6. Web Server (Apache)

Salah satu komponen penting di dalam *website* adalah *web server*. *Web server* berfungsi sebagai penerima request dari browser yang kemudian memberikan tanggap dengan mengirimkan halaman situs web dalam bentuk dokumen HTML. Apache adalah *web server* yang cukup populer sejak dulu. Selain memberikan performa yang andal, Apache juga mempunyai beberapa fitur canggih lain yang mudah digunakan. Jadi tentu saja akan membuat *website* Anda lebih powerful.

Apache menawarkan konsep *based on process* yaitu jika ada pekerjaan baru membuat proses. Apache memiliki sistem pengaturan yang baik karena bisa mencakup pengaturan tingkat lanjut yang akan membuat *website* Anda lebih menarik. Apache juga memiliki masa pengguna yang cukup lama sehingga dokumentasi akan lebih banyak dan mendukung berbagai macam operasi sistem. Apache juga menyajikan konten statis menggunakan metode konvensional dan memproses konten dinamis secara asli di dalam *web server* itu sendiri. Melalui berbagai macam teknologi ini Apache masih tetap digunakan sebagai *web server* dan berhasil menjadi *web server* paling populer sampai dengan saat ini.

[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

4.1. Analisis Sistem

Pada bab ini akan dijelaskan mengenai tahapan dalam membangun aplikasi sistem *marketplace* VEGGO yaitu analisis dari sistem yang akan dibangun. Hal tersebut dijelaskan ke dalam dua bagian, definisi umum aplikasi dan analisis kebutuhan.

4.1.1. Definisi Umum Aplikasi

Secara umum, aplikasi *marketplace* VEGGO merupakan sistem penjualan bahan pangan berbasis *website* yang digunakan menjual sayur organik langsung dari petani dan beberapa bahan makanan. Sistem ini memiliki empat *view* atau empat sisi *user* yaitu pembeli, reseller, petani, dan penjual.

4.1.2. Analisis Kebutuhan

Dalam aplikasi ini, terdapat fungsi-fungsi yang harus dipenuhi oleh sistem. Fungsi-fungsi yang harus dipenuhi tersebut adalah hasil diskusi antara *project manager* dengan *client*. Kebutuhan ini terbagi ke dalam dua jenis, yakni kebutuhan fungsional dan kebutuhan non-fungsional

4.1.2.1. Kebutuhan Fungsional

Kebutuhan fungsional pada aplikasi ini menjelaskan bagaimana sistem ini bekerja yang akan dijelaskan pada Tabel 4.1

Tabel 4.1 Kebutuhan Fungsional Pembeli

Kode	Deskripsi Kebutuhan
F-001	Memilih Tanggal Pengiriman
F-002	Melihat Etalase Barang
F-003	Menambahkan Barang ke Keranjang
F-004	Menghapus Barang per Item
F-005	*Checkout* Keranjang
F-006	Konfirmasi Pembelian
F-007	Melihat Daftar Transaksi
F-008	Mengupload Bukti Pembayaran
F-009	Mengubah Profil Akun
F-0010	Mengubah Alamat Pembeli
F-011	Melihat Riwayat Pembelian
F-012	Konfirmasi Barang Telah Sampai

Untuk penjelasan dari masing-masing kebutuhan fungsional dapat dilihat pada penjelasan berikut:

F-001. Memilih Tanggal Pengiriman

Pembeli harus memilih tanggal pengiriman terlebih dahulu.

F-002. Melihat Etalase Barang

Pembeli yang telah memilih tanggal pengiriman dapat langsung memilih produk di etalase.

F-003. Menambahkan Barang ke Keranjang

Pembeli dapat menambahkan barang ke keranjang (membeli)

F-004. Menghapus Barang per Item

Pembeli dapat menghapus barang dari keranjang.

F-005. *Checkout* Keranjang

Pembeli dapat melihat keranjang pembelian yang menampilkan barang apa saja yang telah dimasukkan keranjang beserta total dan rincian harganya, pembeli juga dapat menghapus barang per item.

F-006. Konfirmasi Pembelian

Pembeli mengkonfirmasi detail transaksinya seperti (alamat pengiriman, rincian belanja, total belanjaan).

F-007. Melihat Daftar Transaksi

Pembeli dapat melihat daftar transaksi yang telah dilakukan (belum dibayar, dalam proses, selesai, dibatalkan).

F-008. *Mengupload* Bukti Pembayaran

Pembeli dapat *mengupload* bukti pembayaran berupa bukti transfer.

F-009. Mengubah Profil Akun

Pembeli dapat mengubah profilnya (no hp, nama pembeli).

F-010. Mengubah Alamat Pembeli

Pembeli dapat mengubah alamat yang menjadi tempat tujuan barang tersebut dikirim.

F-011. Melihat Riwayat Pembelian

Pembeli dapat melihat barang apa saja yang ia telah beli semenjak menggunakan aplikasi ini di halaman transaksi.

F-012. Konfirmasi Barang Telah Sampai

Pembeli dapat mengkonfirmasi ke VEGGO bahwa barang yang dibeli telah sampai.

Tabel 4.2 Kebutuhan Fungsional Reseller

Kode	Deskripsi Kebutuhan
F-013	Memilih Tanggal Pengiriman
F-014	Melihat Etalase Barang
F-015	Menambahkan Barang ke Keranjang
F-016	Mengisi Data Diri Pembeli Offline
F-017	Menghapus Pembeli Offline Beserta Barangnya
F-018	*Checkout* Keranjang
F-019	Konfirmasi Pembelian

F-020	Melihat Daftar Transaksi
F-021	Mengupload Bukti Pembayaran
F-022	Mengubah Profil Reseller
F-023	Mengubah Alamat Reseller
F-024	Melihat Riwayat Belanjaan
F-025	Konfirmasi Barang Telah Sampai

F-013. Memilih Tanggal Pengiriman

Reseller harus memilih tanggal pengiriman terlebih dahulu.

F-014. Melihat Etalase Barang

Reseller yang telah memilih tanggal pengiriman dapat langsung memilih produk di etalase.

F-015. Menambahkan Barang ke Keranjang

Reseller dapat menambahkan barang ke keranjang (membeli).

F-016. Mengisi Data Diri Pembeli Offline

Reseller dapat mengisi data diri pembeli offline dan menyimpan pembelian *offline* orang pertama sampai seterusnya.

F-017. Menghapus Pembeli Beserta Barangnya

Reseller dapat menghapus pembeli *offline* beserta barangnya.

F-018. *Checkout* Keranjang

Reseller dapat melihat keranjang pembelian yang menampilkan siapa saja,

barang apa saja yang telah dimasukkan keranjang beserta total dan rincian harganya, pembeli juga dapat menghapus barang per item.

F-019. Konfirmasi Pembelian

Reseller mengkonfirmasi detail transaksinya seperti (alamat pengiriman, rincian belanja, total belanjaan).

F-020. Melihat Daftar Transaksi

Reseller dapat melihat daftar transaksi yang telah dilakukan (belum dibayar, dalam proses, selesai, dibatalkan).

F-021. Menguploat Bukti Pembayaran

Reseller dapat mengupload bukti pembayaran berupa bukti transfer.

F-022. Mengubah Profil Reseller

Reseller dapat mengubah profilnya (nomor HP, nama pembeli).

F-023. Mengubah Alamat Reseller

Reseller dapat mengubah alamat yang menjadi tempat tujuan barang tersebut dikirim.

F-024. Melihat Riwayat Pembelian

Reseller dapat melihat barang apa saja yang ia telah beli semenjak menggunakan aplikasi ini di halaman transaksi.

F-025. Konfirmasi Barang Telah Sampai

Reseller dapat mengkonfirmasi ke VEGGO bahwa barang yang dibeli telah samapai.

Tabel 4.3 Kebutuhan Fungsional Penjual

Kode	Deskripsi Kebutuhan
F-026	*Login* ke Penjual
F-027	Melihat Kategori Produk
F-028	Menambah Kategori Produk
F-029	Mengubah Kategori Produk
F-030	Melihat Produk
F-031	Menambah Produk
F-032	Mengubah Produk
F-033	Melihat Paket
F-034	Menambah Paket
F-035	Mengubah Paket
F-036	Membuka Tanggal Pengiriman
F-037	Menutup Tanggal Pengiriman
F-038	Memlih Produk Pada Tanggal Pengiriman
F-039	Melihat Daftar *Pre-Order* Pembeli
F-040	Melihat Detail *Order* Pembeli
F-041	Mengubah Status Pembayaran Pembeli

F-042	Melakukan Akumulasi *Pre-Order*
F-043	Membatalkan *Order* Pembeli
F-044	Melihat Daftar *Order* yang Dibatalkan
F-045	Mengirimkan *Order* ke Petani
F-046	Melakukan Konfirmasi *Order* ke Petani
F-047	Melakukan Klaim *Order* ke Petani
F-048	Melihat Daftar Klaim
F-049	Melihat Detail Klaim
F-050	Melihat Daftar Proses *Order* Pembeli
F-051	Melakukan Finalisasi Pengiriman ke Pembeli
F-052	Melihat Daftar *Order* yang Telah Selesai
F-053	Melihat Inventaris Produk
F-054	Melihat Inventaris Paket
F-055	Menambah Manual Stok Produk

F-056	Menambah Manual Stok Paket
F-057	Melihat *Report* Harian
F-058	Melihat *Report* Bulanan
F-059	Melihat *Report* Reseller
F-060	Melihat Daftar Tagihan ke Petani

Untuk penjelasan dari masing-masing kebutuhan fungsional dari Penjual dapat dilihat pada penjelasan berikut:

F-026. *Login* ke Penjual
Penjual dapat login ke akunnya sesuai dengan rolenya.

F-027. Melihat Kategori Produk
Penjual dapat melihat semua Kategori Produknya.

F-028. Menambah Kategori Produk
Penjual dapat menambah kategori produknya.

F-029. Mengubah Kategori Produk
Penjual dapat mengubah Kategori Produknya.

F-030. Melihat Produk
Penjual dapat melihat semua Produknya.

F-031. Menambah Produk
Penjual dapat menambah Produknya sendiri.

F-032. Mengubah Produk

Penjual dapat mengubah Produknya sendiri.

F-033. Melihat Paket

Penjual dapat melihat semua Paketnya.

F-034. Menambah Paket

Penjual dapat menambah Paketnya sendiri.

F-035. Mengubah Paket

Penjual dapat mengubah Paketnya sendiri.

F-036. Membuka Tanggal Pengiriman

Penjual dapat membuka tanggal pengiriman.

F-037. Menutup Tanggal Pengiriman

Penjual dapat menutup tanggal pengiriman.

F-038. Memlih Produk pada Tanggal Pengiriman

Penjual dapat memilih produk mana yang akan ditampilkan di etalase pembeli pada tanggal pengiriman tertentu.

F-039. Melihat Daftar *Pre-Order* Pembeli

Penjual dapat melihat daftar *Pre-orde*r pembeli.

F-040. Melihat Detail *Order* Pembeli

Penjual dapat melihat detail *order* Pembeli.

F-041. Mengubah Status Pembayaran Pembeli
Penjual dapat mengubah status pembayaran Pembeli (Lunas, Belum Dibayar, Tunggu Konfirmasi, Gagal),

F-042. Melakukan Akumulasi *Pre-Order*
Penjual dapat melakukan akumulasi *Pre-order*.

F-043. Membatalkan *Order* Pembeli
Penjual dapat membatalkan order Pembeli.

F-044. Melihat Daftar *Order* yang Dibatalkan
Penjual dapat melihat daftar order yang dibatalkan.

F-045. Mengirimkan *Order* ke Petani
Penjual dapat mengirimkan order barang ke Petani.

F-046. Melakukan Konfirmasi Order ke Petani
Penjual dapat melakukan konfirmasi order barang ke Petani.

F-047. Melakukan Klaim *Order* ke Petani
Penjual dapat melakukan klaim order barang ke Petani.

F-048. Melihat Daftar Klaim
Penjual dapat melihat daftar klaim ke Petani.

F-049. Melihat Detail Klaim
Penjual dapat melihat detail klaim ke Petani.

F-050. Melihat Daftar Proses *Order* Pembeli
Penjual dapat melihat daftar proses order Pembeli.

F-051. Melakukan Finalisasi Pengiriman ke Pembeli
Penjual dapat melakukan finalisasi pengiriman ke Pembeli.

F-052. Melihat daftar *order* yang selesai
Penjual dapat melihat daftar order yang telah dilakukan.

F-053. Melihat Inventaris Produk
Penjual dapat melihat inventaris Produk.

F-054. Melihat Inventaris Paket
Penjual dapat melihat inventaris Paket

F-055. Menambah Manual Stok Produk
Penjual dapat menambah manual produknya.

F-056. Menambah Manual Stok Paket
Penjual dapat menambah manual stok Paketnya.

F-057. Melihat *Report* Harian
Penjual dapat melihat report dalam harian.

F-058. Melihat *Report* Bulanan
Penjual dapat melihat report dalam bulanan.

F-059. Melihat *Report* Reseller
Penjual dapat melihat report ke Reseller.

F-060. Melihat Daftar Tagihan ke Petani
Penjual dapat melihat daftar tagihan ke Petani.

Tabel 4.4 Kebutuhan Fungsional Petani

Kode	Deskripsi Kebutuhan
F-061	Melakukan *Login* ke Petani
F-062	Melihat Daftar *Order* ke Petani
F-063	Melakukan Pengiriman *Order* ke Penjual
F-064	Melihat Daftar Klaim
F-065	Melihat Detail Klaim
F-066	Mengatur Ketersediaan Produk yang Dimilikinya
F-067	Melihat Tagihan Dari Penjual

Untuk penjelasan dari masing-masing kebutuhan fungsional dari Petani dapat dilihat pada penjelasan berikut:

F-061. Melakukan *Login* ke Petani

Petani dapat *login* ke akunnya sesuai dengan rolenya.

F-062. Melihat Daftar *Order* ke Petani

Petani dapat melihat daftar *order* dari Penjual.

F-063. Melakukan Pengiriman *Order* ke Penjual

Petani dapat mengirimkan *order* barang ke Penjual.

F-064. Melihat Daftar Klaim

Petani dapat melihat daftar klaim dari Penjual.

F-065. Melihat Detail Klaim

Petani dapat melihat detail klaim dari penjual.

F-066. Mengatur Ketersediaan Produk yang Dimiliki

Petani dapat mengatur ketersediaan produknya.

F-067. Melihat Tagihan Dari Penjual

Petani dapat melihat daftar tagihan dari penjual.

4.1.2.2. Kebutuhan Non Fungsional

Kebutuhan non-fungsional adalah kebutuhan pengguna untuk mendefinisikan bagaimana batasan dan karakteristik dari sebuah sistem yang dibangun. Kebutuhan non-fungsional dari aplikasi ini terdapat pada Tabel 4.5.

Tabel 4.5 Kebutuhan Non-Fungsional

Kode	Deskripsi Kebutuhan
NF-001	Dapat diakses oleh semua *web-browser*

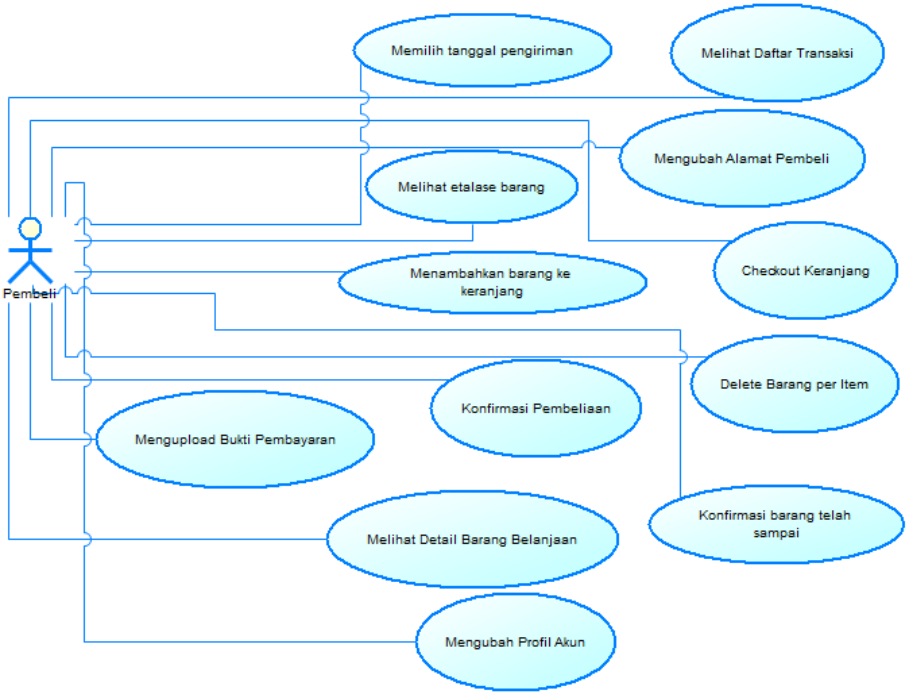
NF-002	Sistem dapat diakses 24 jam sehari
NF-003	Sistem dapat diakses oleh semua perangkat yang memiliki *web-browser* yang mendukung HTML 5.0
NF-004	Sistem memiliki *feedback* yang baik
NF-005	Sistem harus dapat memberikan hak akses data kepada pengguna yang berhak.

4.2. Perancangan Sistem

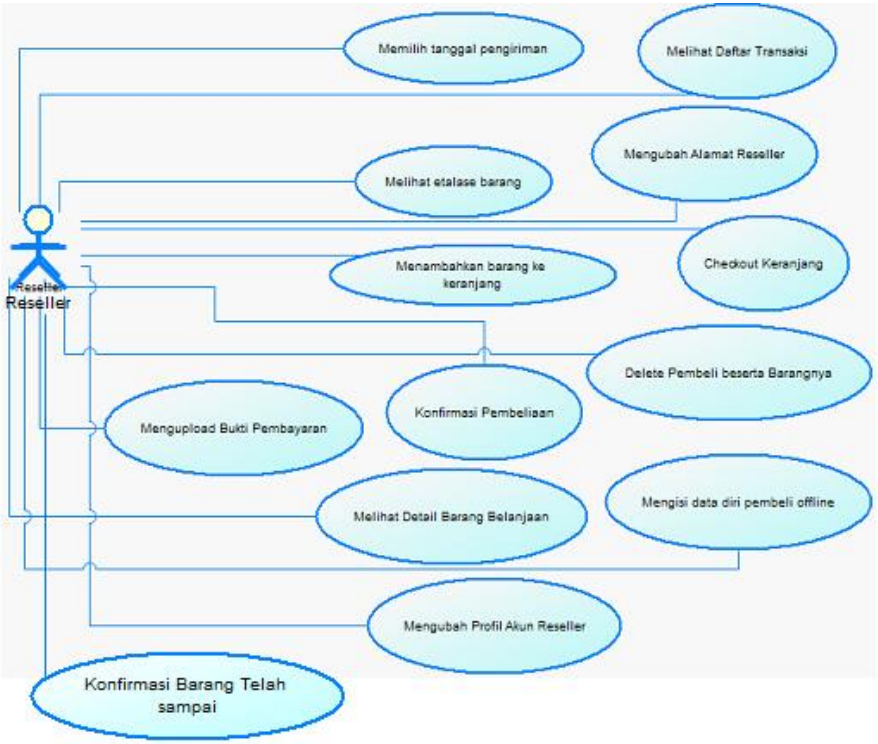
4.2.1. Desain Sistem

Desain sistem digunakan untuk mengetahui jalannya proses bisnis pada suatu aplikasi sehingga pengembangan dan pemeliharaan aplikasi dapat dengan mudah dilakukan. Desain sistem yang digunakan adalah *Use Case Diagram, Context Diagram,* dan *Activity Diagram.*

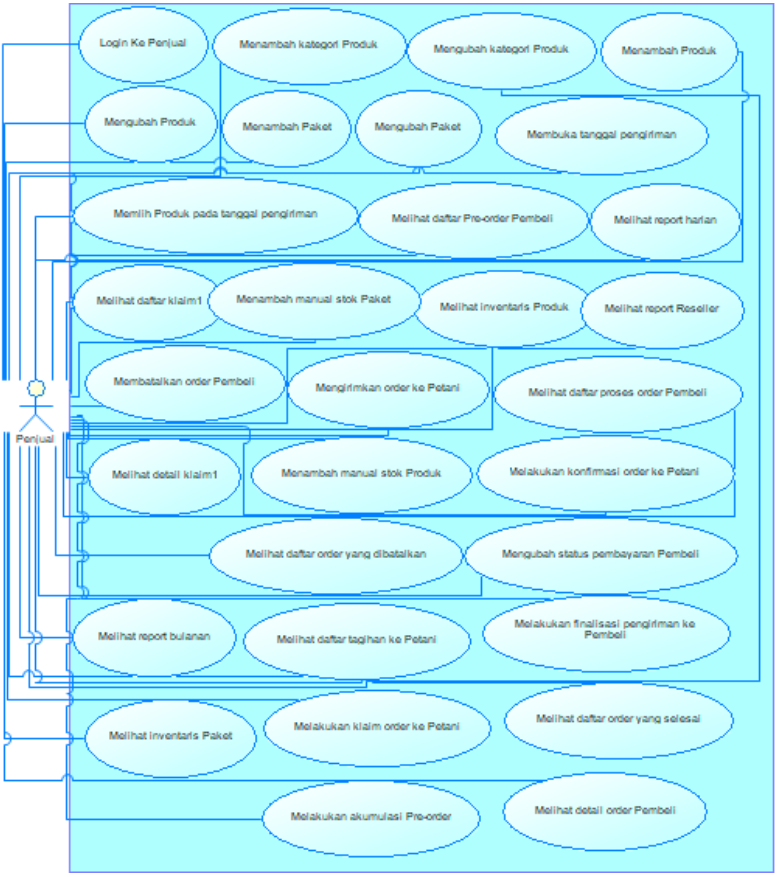
Gambar 4.1. merupakan *Use Case Diagram* yang menunjukkan proses bisnis apa saja dan siapa saja yang terlibat dalam proses tersebut. Gambar 4.5 menunjukkan *Context Diagram* yang menjelaskan alur data (*data flow*) antara sistem dengan *stakeholder*. Terakhir, *Activity Diagram* menunjukkan bagaimana sistem berinteraksi dengan *stakeholder*.



Gambar 4.1 Use-Case Diagram Pembeli



Gambar 4.2 Use-Case Diagram Reseller



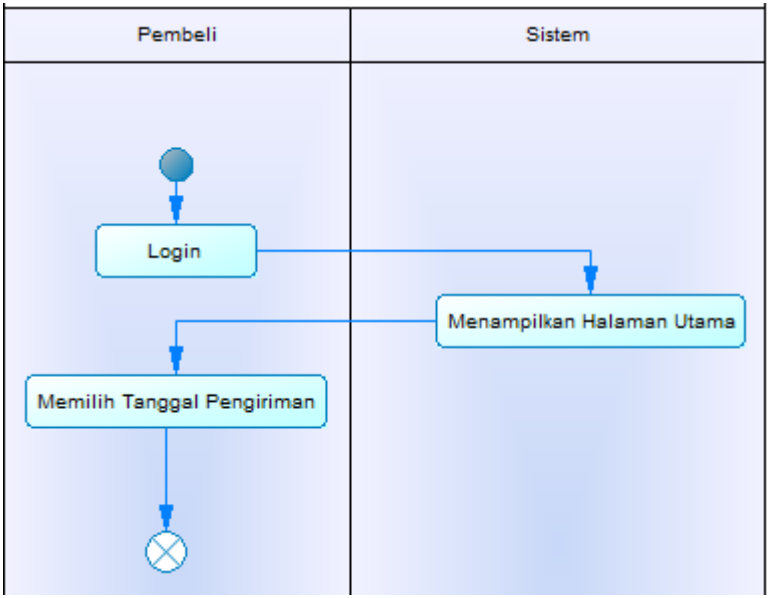
Gambar 4.3. Use-Case Diagram Penjual

Gambar 4.4. Use-Case Diagram Petani

F-001. Memilih Tanggal Pengiriman

Sebelum memilih barang pembeli harus memilih tanggal pengiriman.

Gambar 4.5. di bawah ini merupakan diagram aktivitas yang menunjukkan alur memilih tanggal pengiriman

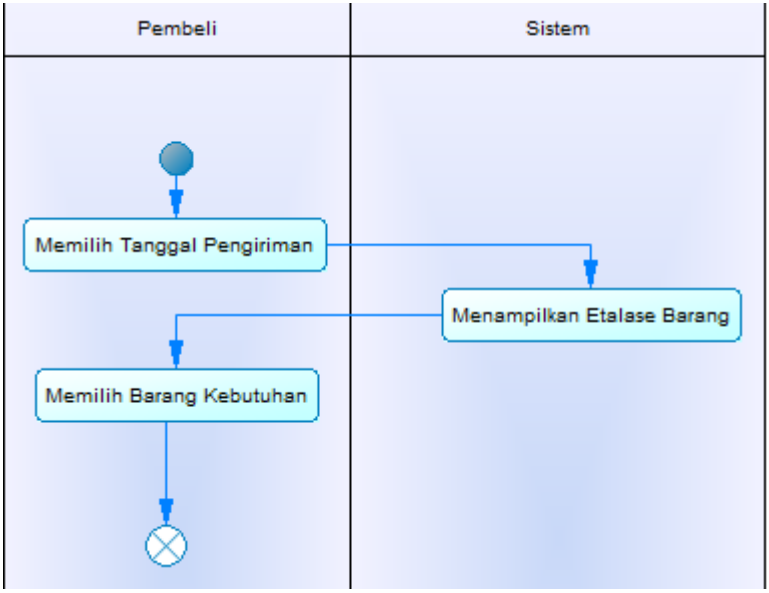


Gambar 4.5. Activity Diagram Memilih Tanggal Pengiriman

F-002. Melihat Etalase Barang

Setelah memilih tanggal pengiriman Pembeli dapat memilih barang belanjaan.

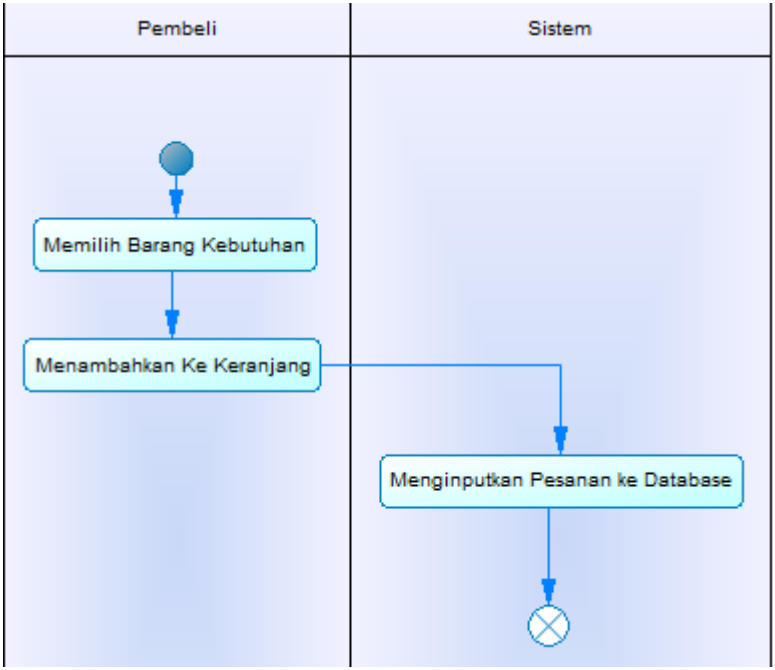
Gambar 4.6. di bawah ini merupakan diagram aktivitas yang menunjukkan alur memilih barang di halaman etalase barang



Gambar 4.6. Activity Diagram Melihat Etalase Barang

F-003. Menambahkan Barang ke Keranjang

Pembeli dapat menambahkan barang ke keranjang Gambar 4.7. di bawah ini merupakan diagram aktivitas yang menunjukkan alur menambahkan barang ke keranjang

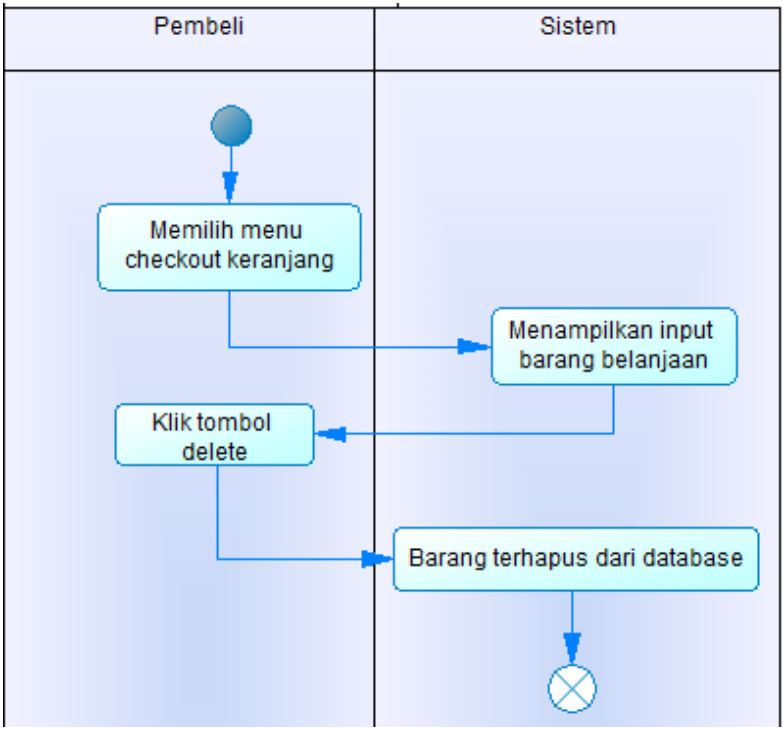


Gambar 4.7. Activity Diagram Menambahkan Barang ke Keranjang

F-004. Menghapus Barang per Item

Pembeli dapat menghapus barang per item.

Gambar 4.8. di bawah ini merupakan diagram aktivitas yang menunjukkan alur menghapus barang per item

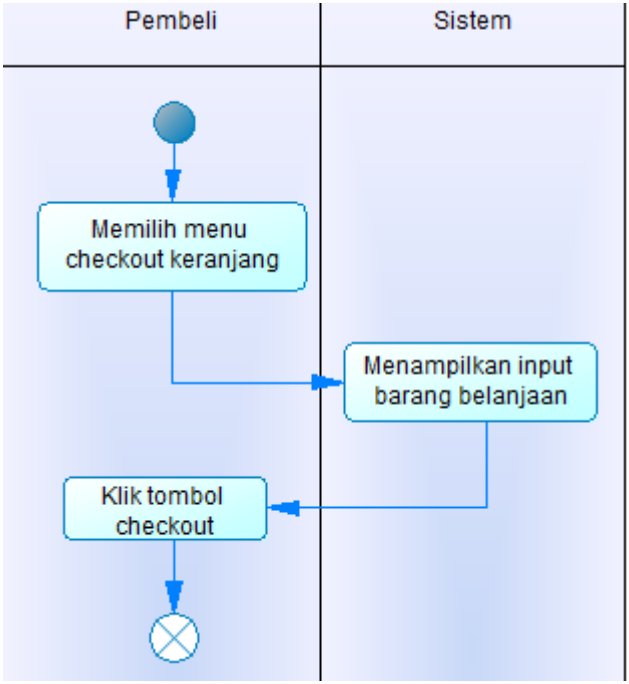


Gambar 4.8. Activity Diagram Menghapus Barang per Item

F-005. *Checkout* Keranjang

Pembeli dapat melakukan checkout keranjang jika barang yang dipilihnya sudah tepat

Gambar 4.9. di bawah ini merupakan diagram aktivitas yang menunjukkan alur checkout keranjang

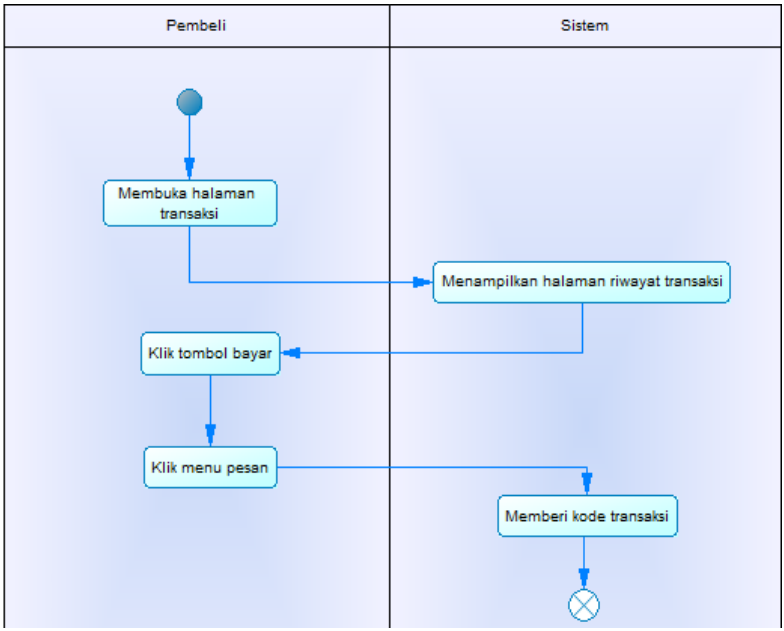


Gambar 4.9. Activity Diagram Checkout Keranjang

F-006. Konfirmasi Pembelian

Setelah checkout keranjang, pembeli dapat melakukan konfirmasi pembelian

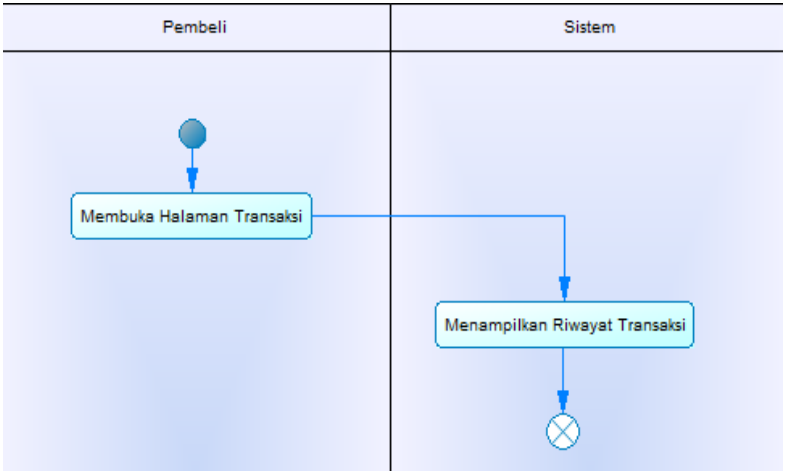
Gambar 4.10. di bawah ini merupakan diagram aktivitas yang menunjukkan alur konfirmasi pembelian



Gambar 4.10. Activity Diagram Konfirmasi Pembelian

F-007. Melihat Daftar Transaksi

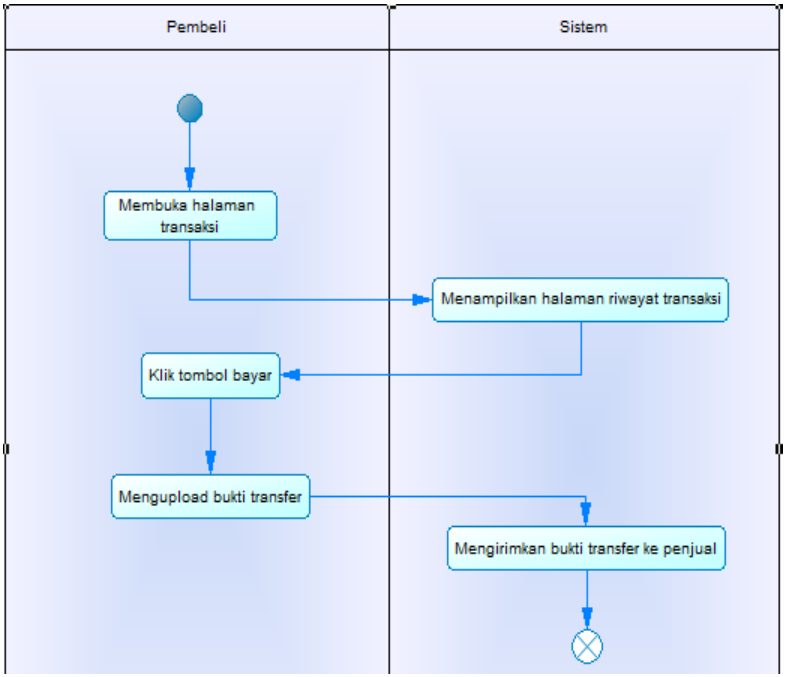
Pembeli dapat melihat daftar transaksi yang belum dibayar hingga transaksi yang sudah selesai. Gambar 4.11. di bawah ini merupakan diagram aktivitas yang menunjukkan alur melihat daftar transaksi.



Gambar 4.11. Activity Diagram melihat Daftar Transaksi

F-008. *Mengupload* Bukti Pembayaran

Pembeli dapat *mengupload* bukti pembayaran. Gambar 4.12. di bawah ini merupakan diagram aktivitas yang menunjukkan alur mengupload bukti pembayaran.

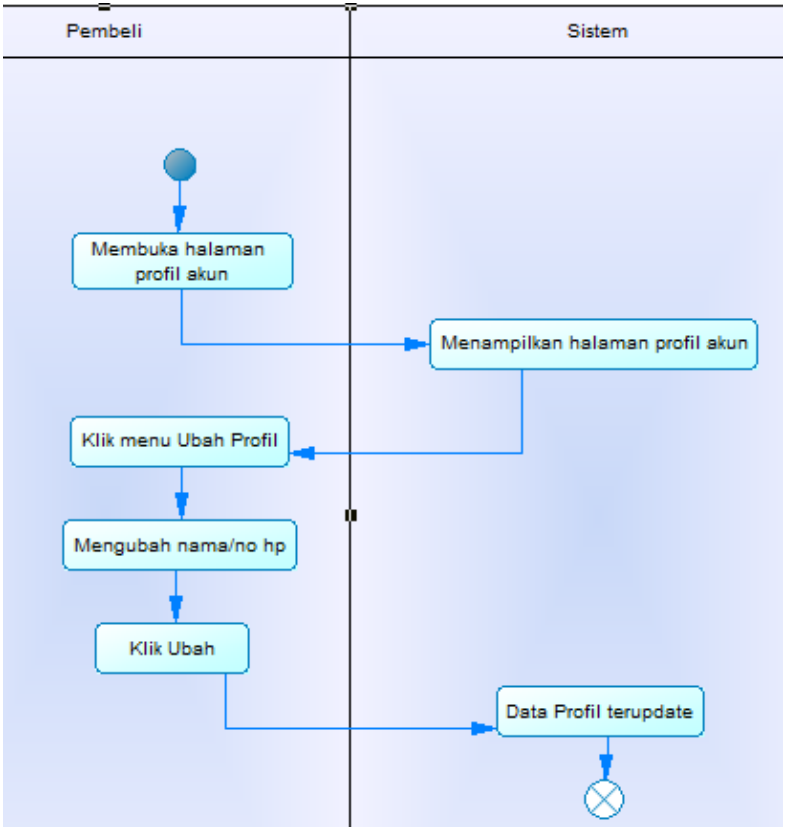


Gambar 4.12. Activity Diagram Mengupload Bukti Pembayaran

F-009. Mengubah Profil Akun

Pembeli dapat mengubah profilnya (nama dan nomor hp)

Gambar 4.13. di bawah ini merupakan diagram aktivitas yang menunjukkan alur mengubah profil pembeli.

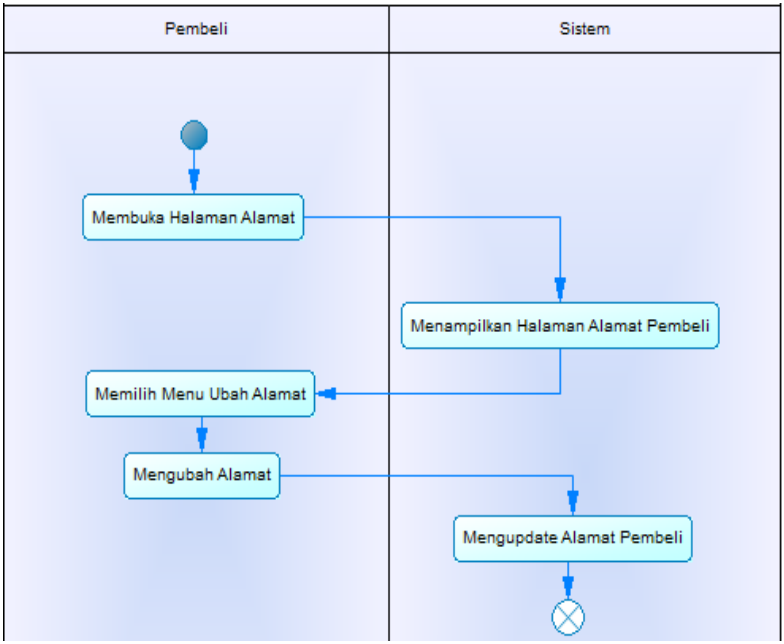


Gambar 4.13. Activity Diagram Mengubah Profil Akun

F-010. Mengubah Alamat Pembeli

Pembeli dapat mengubah alamatnya yang menjadi alamat tujuan pengantaran.

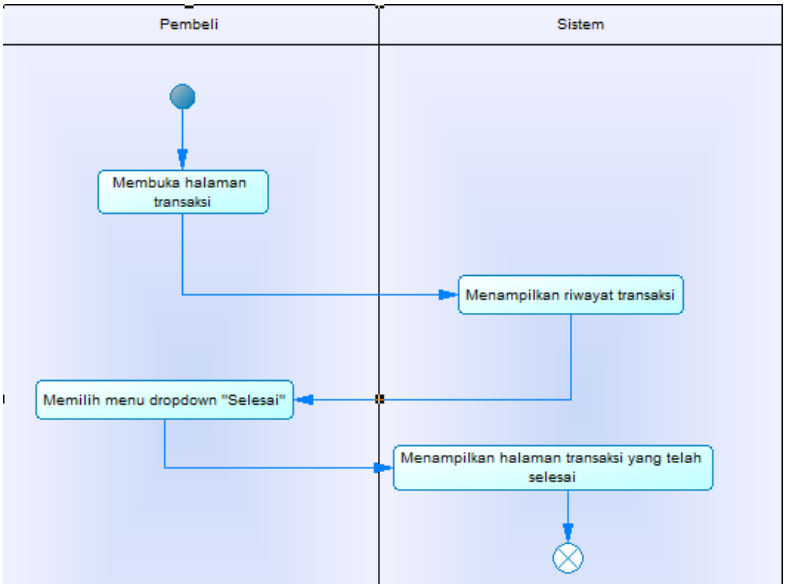
Gambar 4.14. di bawah ini merupakan diagram aktivitas yang menunjukkan alur mengubah alamat pembeli



Gambar 4.14. Activity Diagram Mengubah Alamat Pembeli

F-011. Melihat Riwayat Pembelian

Pembeli dapat melihat riwayat belanjaan yang telah dilakukan sejak menggunakan aplikasi ini. Gambar 4.15. di bawah ini merupakan diagram aktivitas yang menunjukkan alur melihat riwayat belanjaan.

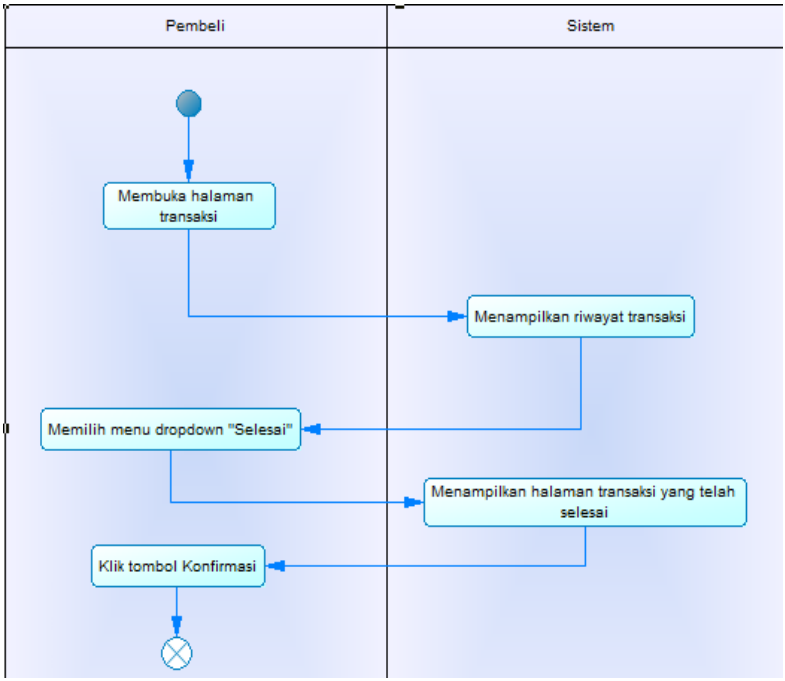


Gambar 4.15. Activity Diagram Melihat Riwayat Pembelian

F-012. Konfirmasi Barang Telah Sampai

Pembeli melakukan konfirmasi barang telah sampai setelah barang sampai.

Gambar 4.16. di bawah ini merupakan diagram aktivitas yang menunjukkan alur konfirmasi barang telah sampai.

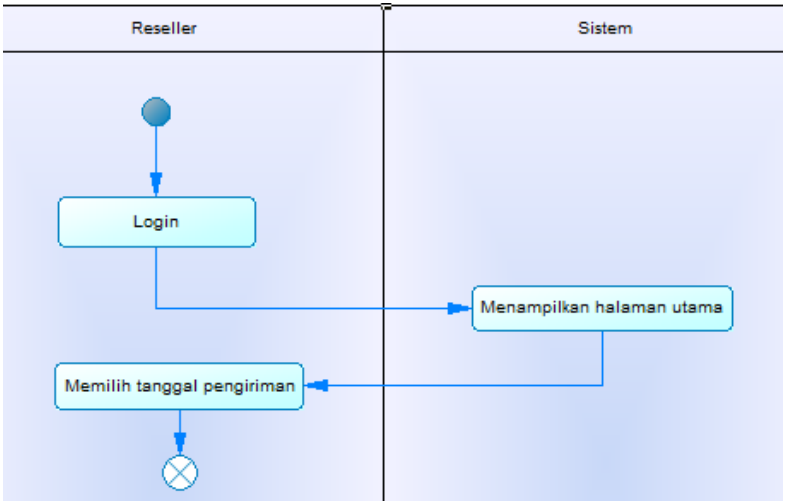


Gambar 4.16. Activity Diagram Konfirmasi Barang Telah Sampai

F-013. Memilih Tanggal Pengiriman

Sebelum memilih barang Reseller harus memilih tanggal pengiriman.

Gambar 4.17. di bawah ini merupakan diagram aktivitas yang menunjukkan alur memilih tanggal pengiriman.

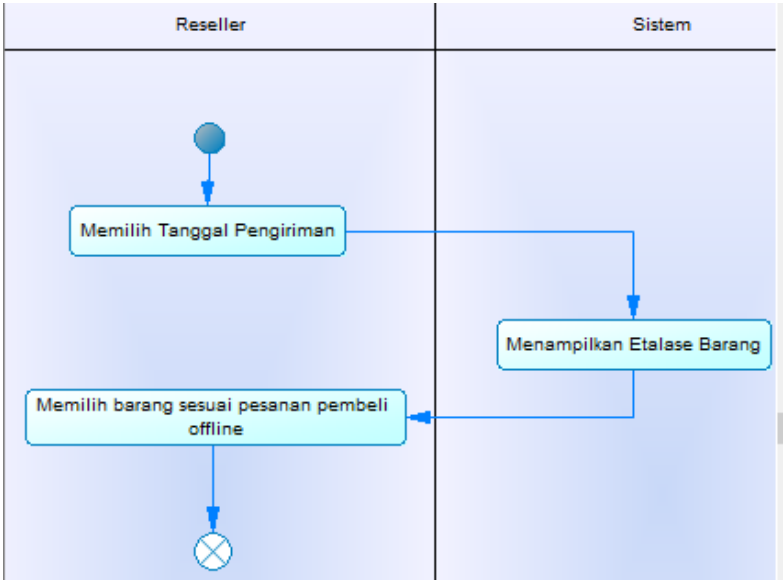


Gambar 4.17. Activity Diagram Memilih Tanggal Pengiriman

F-014. Melihat Etalase Barang

Setelah memilih tanggal pengiriman Reseller dapat memilih barang belanjaan.

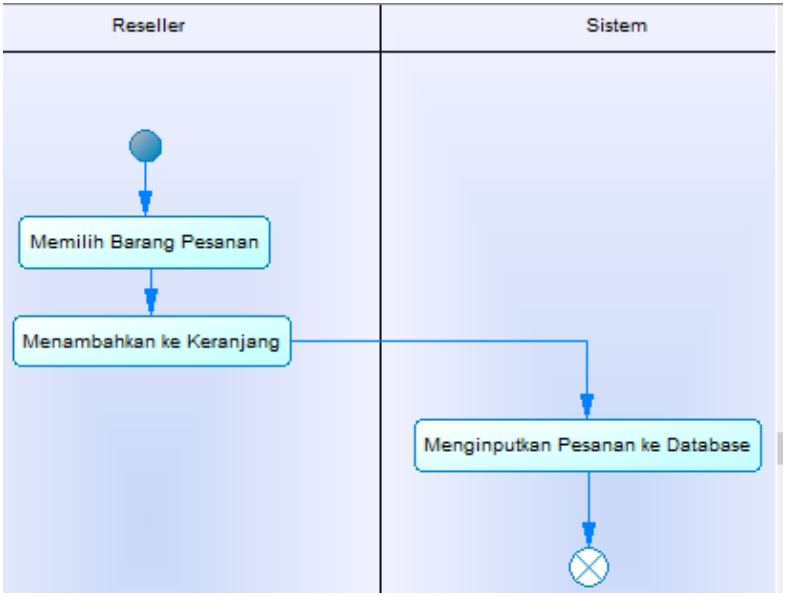
Gambar 4.18. di bawah ini merupakan diagram aktivitas yang menunjukkan alur memilih barang di halaman etalase barang



Gambar 4.18. Activity Diagram Melihat Etalase Barang

F-015. Menambahkan Barang ke Keranjang

Reseller dapat menambahkan barang ke keranjang. Gambar 4.19. di bawah ini merupakan diagram aktivitas yang menunjukkan alur menambahkan barang ke keranjang.

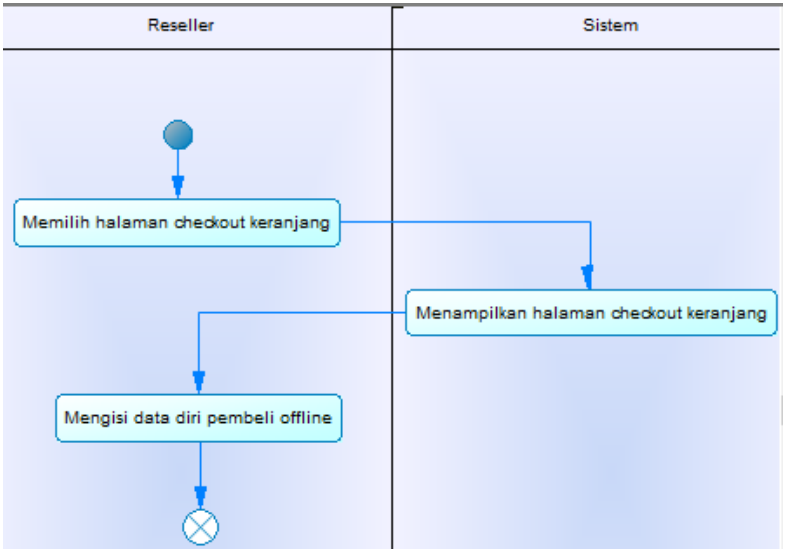


Gambar 4.19. Activity Diagram Menambahkan Barang ke Keranjang

F-016. Mengisi Data Diri Pembeli *Offline*

Setelah menambahkan barang ke keranjang, Reseller harus mengisi data pembeli *offline*.

Gambar 4.20. di bawah ini merupakan diagram aktivitas yang menunjukkan alur mengisi data diri pembeli offline.

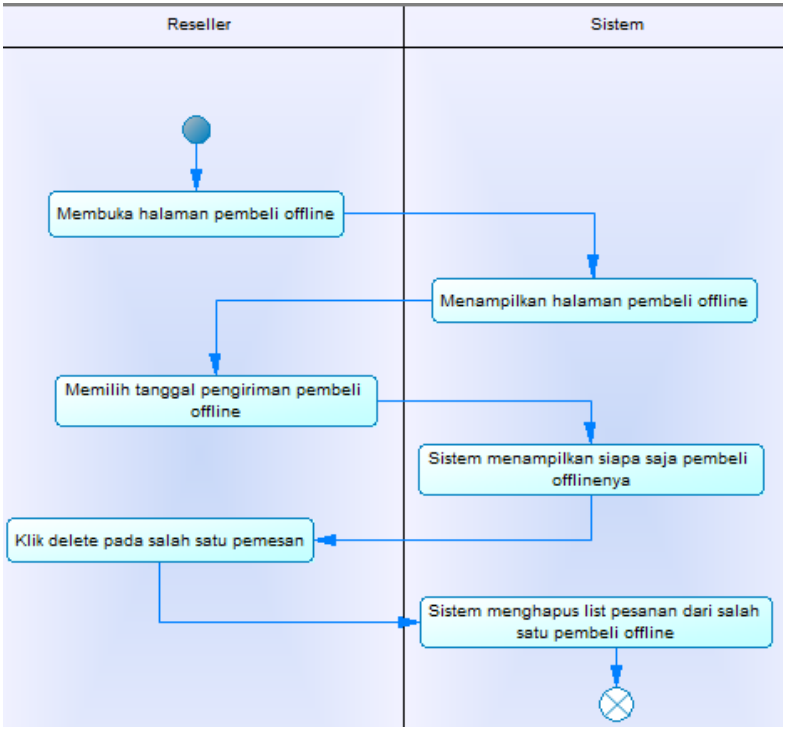


Gambar 4.20. Activity Diagram Mengisi Data Diri Pembeli Offline

F-017. Menghapus Pembeli Beserta Barangnya

Reseller dapat menghapus pembelian jika pembeli *offline* tidak jadi pesan.

Gambar 4.21. di bawah ini merupakan diagram aktivitas yang menunjukkan alur menghapus pembelian

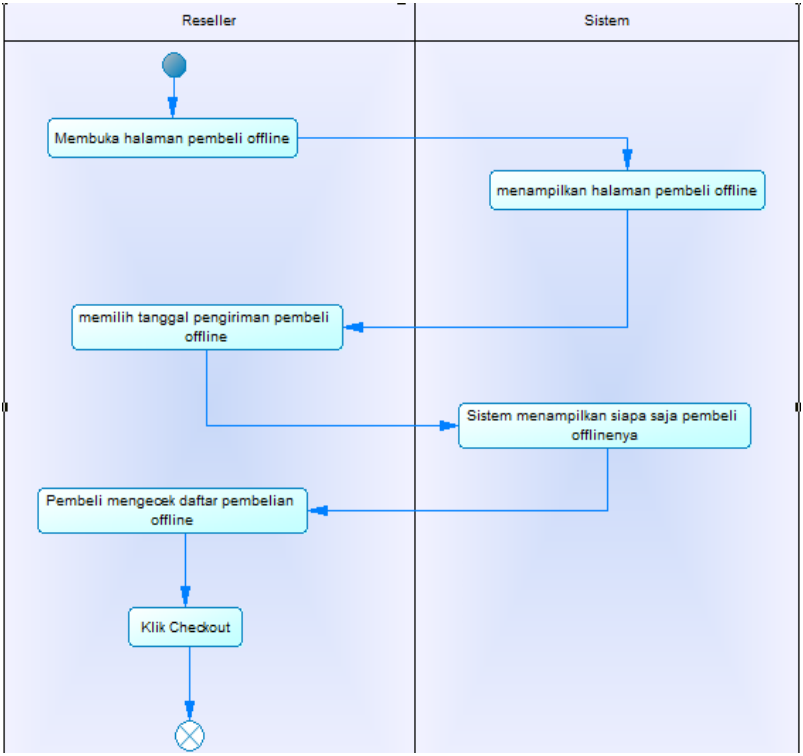


Gambar 4.21. Activity Diagram Menghapus Pembeli Beserta Barangnya

F-018. *Checkout* Keranjang

Reseller dapat melakukan *checkout* keranjang jika barang yang dipilihnya sudah tepat dengan pesanan pembeli *offline*.

Gambar 4.22. di bawah ini merupakan diagram aktivitas yang menunjukkan alur checkout keranjang.

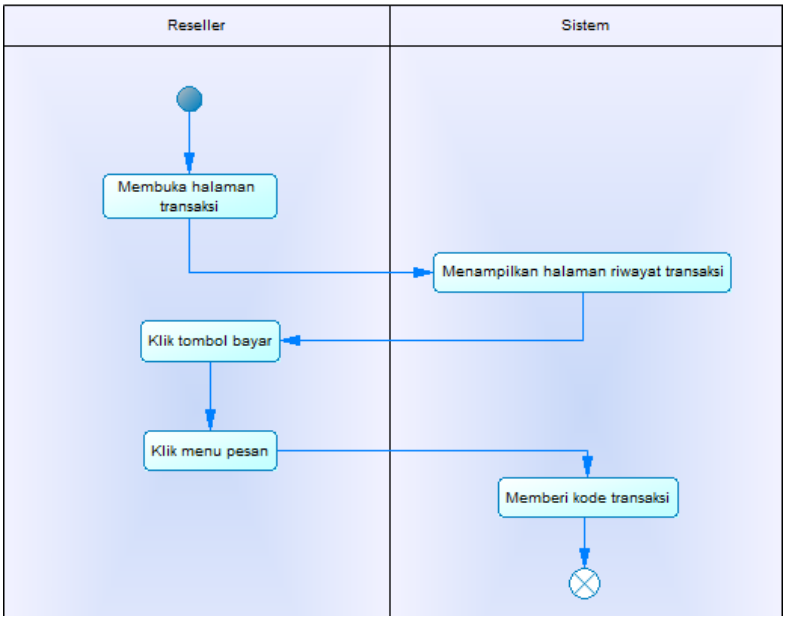


Gambar 4.22. Activity Diagram Checkout Keranjang

F-019. Konfirmasi Pembelian

Setelah *checkout* keranjang, pembeli dapat melakukan konfirmasi pembelian.

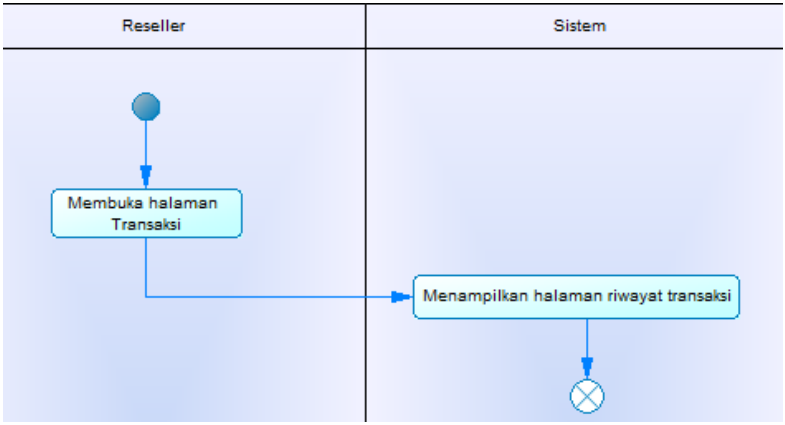
Gambar 4.23. di bawah ini merupakan diagram aktivitas yang menunjukkan alur konfirmasi pembelian.



Gambar 4.23. Activity Diagram Konfirmasi Pembelian

F-020. Melihat Daftar Transaksi

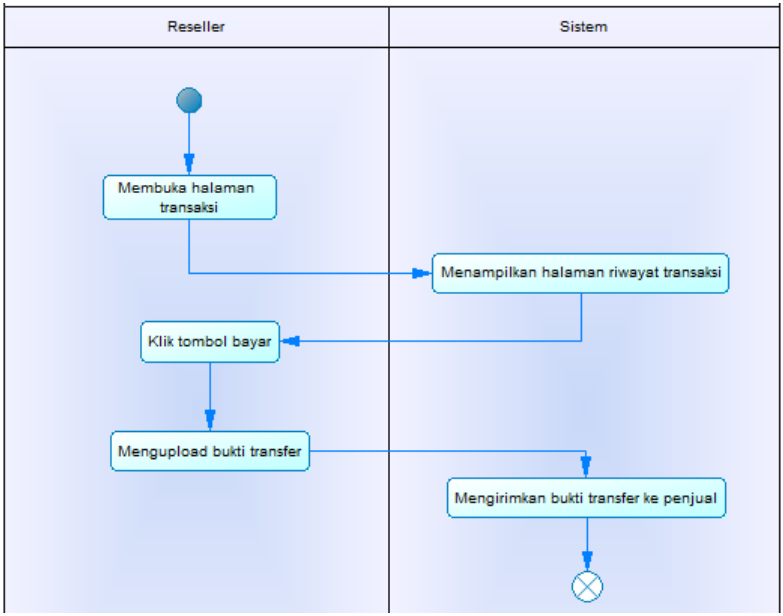
Reseller dapat melihat daftar transaksi yang belum dibayar hingga transaksi yang sudah selesai. Gambar 4.24. di bawah ini merupakan diagram aktivitas yang menunjukkan alur melihat daftar transaksi.



Gambar 4.24. Activity Diagram Melihat Daftar Transaksi

F-021. *Mengupload* Bukti Pembayaran

Reseller dapat *mengupload* bukti pembayaran. Gambar 4.25. di bawah ini merupakan diagram aktivitas yang menunjukkan alur *mengupload* bukti pembayaran.

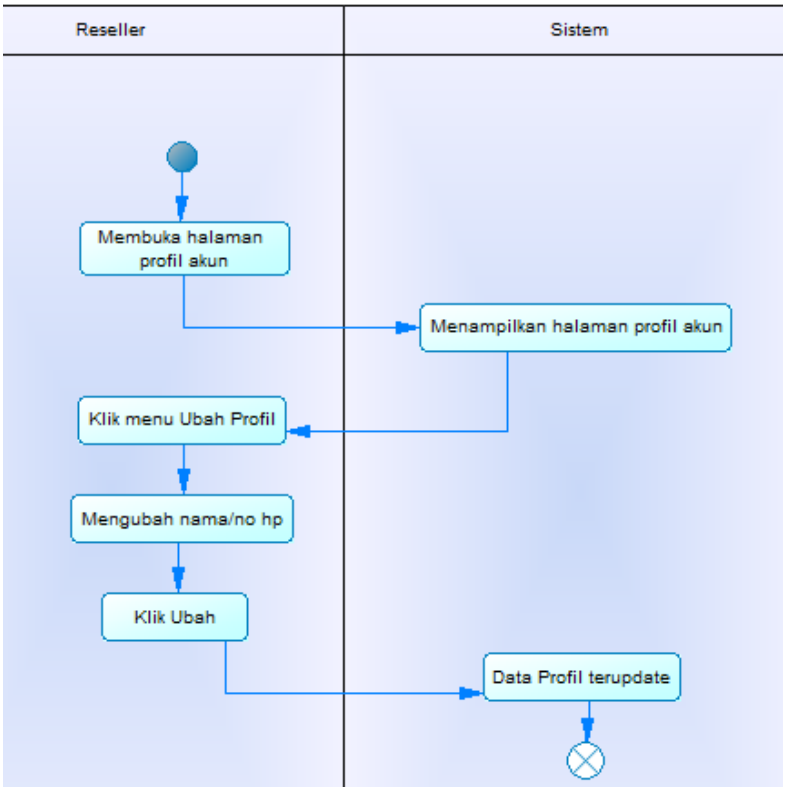


Gambar 4.25. Activity Diagram Mengupload Bukti Pembayaran

F-022. Mengubah Profil Reseller

Reseller dapat mengubah profilnya (nama dan nomor HP).

Gambar 4.26. di bawah ini merupakan diagram aktivitas yang menunjukkan alur mengubah profil reseller.

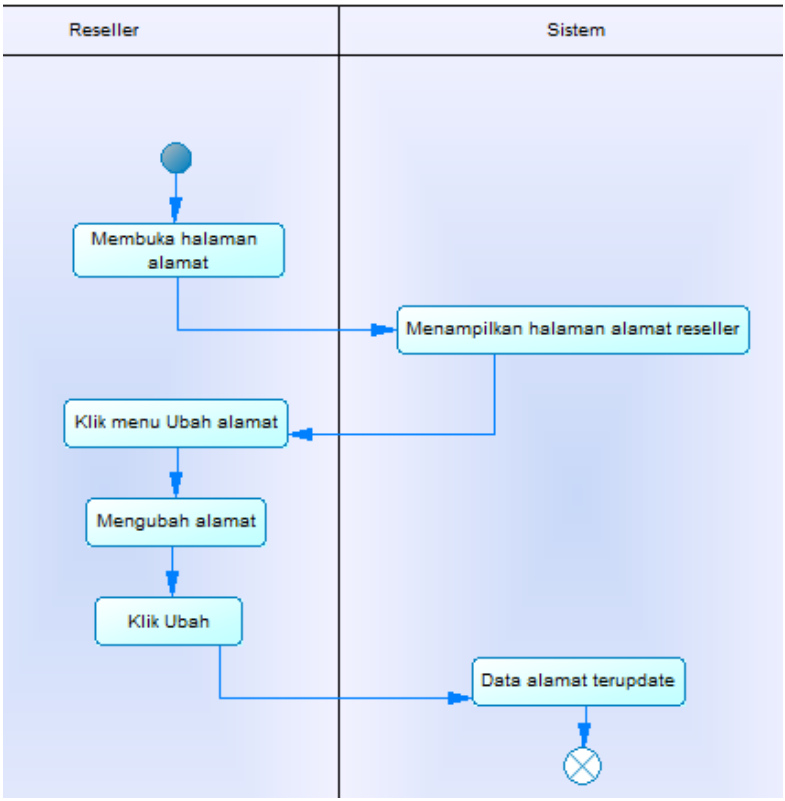


Gambar 4.26. Activity Diagram Mengubah Profil Reseller

F-023. Mengubah Alamat Reseller

Reseller dapat mengubah alamatnya yang menjadi alamat tujuan pengantaran.

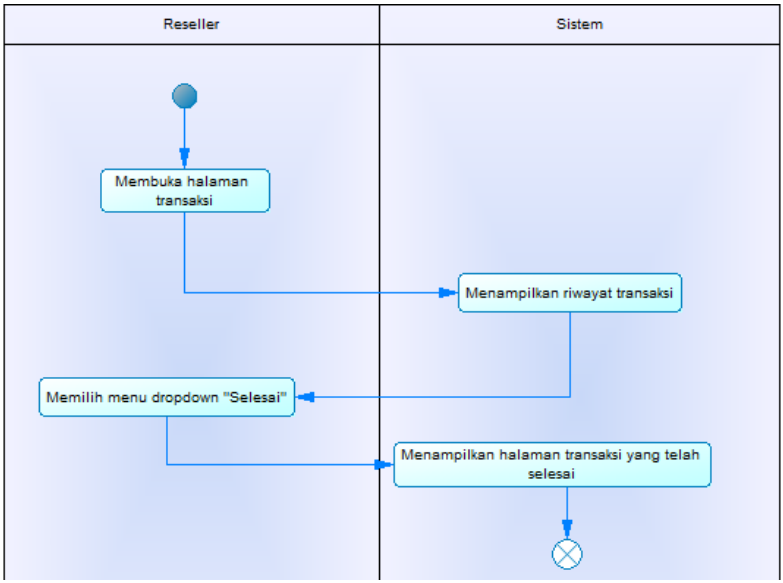
Gambar 4.27. di bawah ini merupakan diagram aktivitas yang menunjukkan alur mengubah alamat reseller



Gambar 4.27. Activity Diagram Mengubah Alamat Reseller

F-024. Melihat Riwayat Pembelian

Reseller dapat melihat riwayat belanjaan yang telah dilakukan sejak menggunakan aplikasi ini. Gambar 4.28. di bawah ini merupakan diagram aktivitas yang menunjukkan alur melihat riwayat belanjaan.

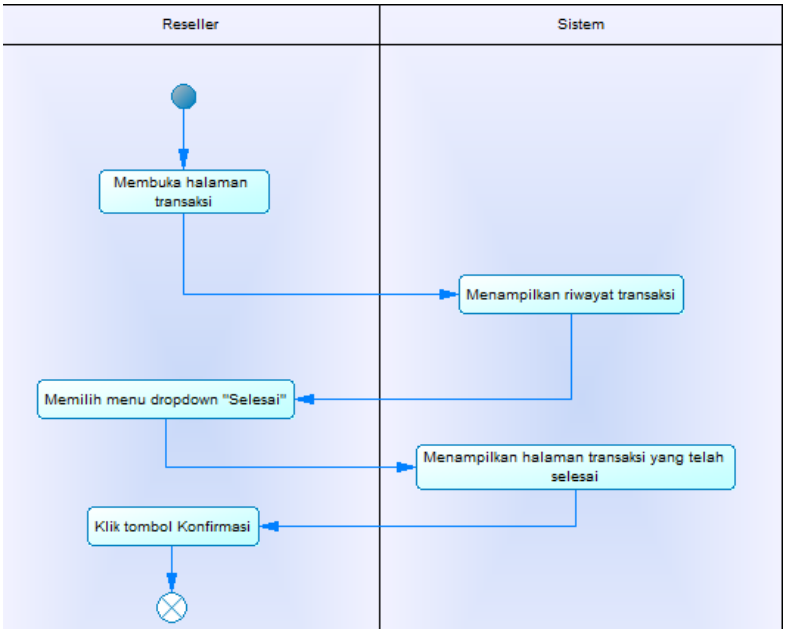


Gambar 4.28. Activity Diagram Melihat Riwayat Pembelian

F-025. Konfirmasi Barang Telah Sampai

Reseller melakukan konfirmasi barang telah sampai setelah barang sampai.

Gambar 4.29. di bawah ini merupakan diagram aktivitas yang menunjukkan alur konfirmasi barang telah sampai.

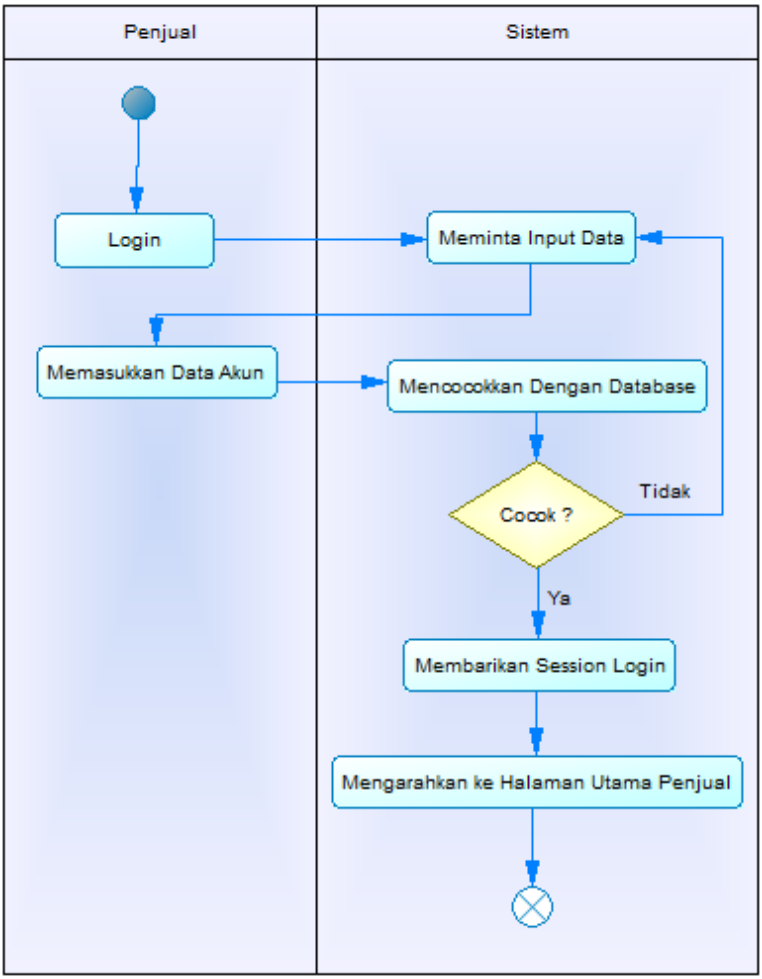


Gambar 4.29. Activity Diagram Konfirmasi Barang Telah Sampai

F-026. Autentikasi Penjual

Penjual melakukan *login* kedalam aplikasi untuk memastikan keaslian hak akses.

Gambar 4.30. di bawah ini merupakan diagram aktivitas yang menunjukkan alur autentikasi

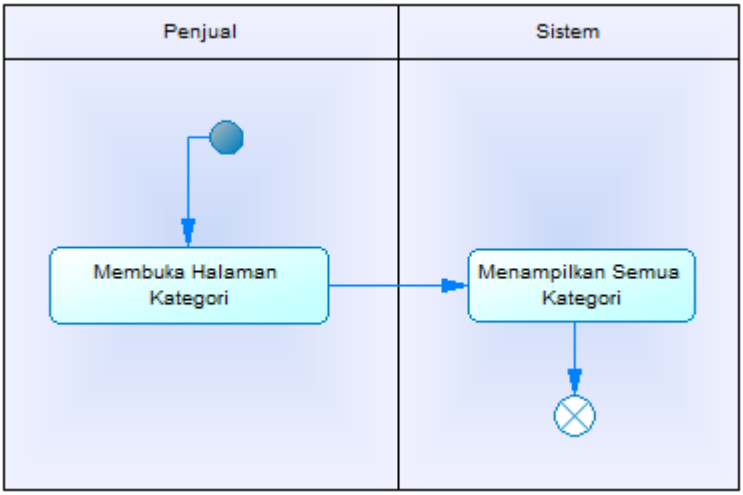


Gambar 4.30. Activity Diagram Autentikasi Penjual

F-027. Melihat Kategori Produk

Penjual dapat melihat kategori produknya. Gambar 4.31 di bawah ini merupakan diagram

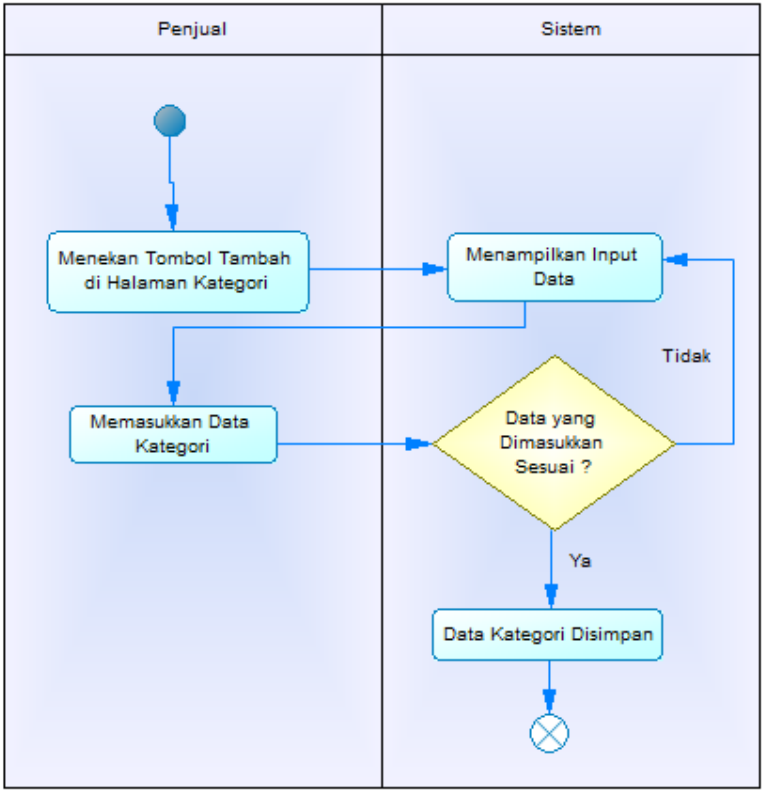
aktivitas yang menunjukkan alur untuk Melihat Kategori Produk.



Gambar 4.31. Activity Diagram Melihat Kategori Produk

F-028. Menambah Kategori Produk

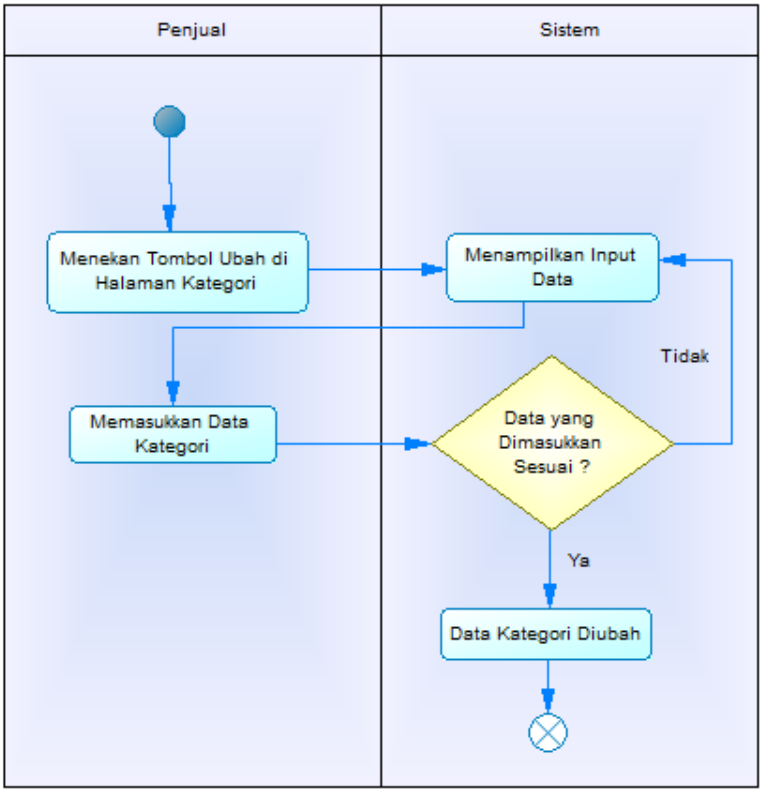
Ketika melihat Halaman Kategori, Penjual dapat Menambah Kategori Produk. Gambar 4.32 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Menambah Kategori Produk.



Gambar 4.32. Activity Diagram Menambah Kategori Produk

F-029. Mengubah Kategori Produk

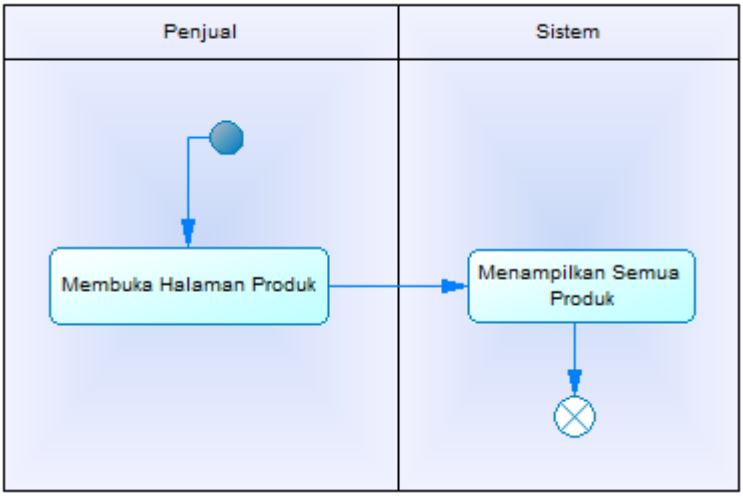
Ketika melihat halaman kategori, Penjual dapat Mengubah Kategori Produk. Gambar 4.33 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Mengubah Kategori Produk.



Gambar 4.33. Activity Diagram Mengubah Kategori Produk

F-030. Melihat Produk

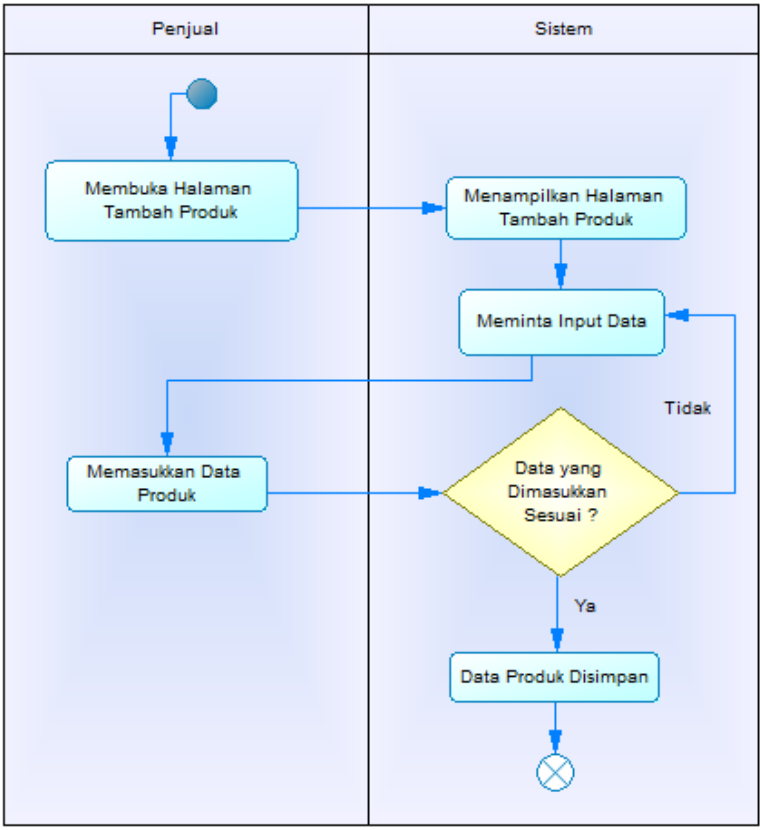
Penjual dapat Melihat Produknya. Gambar 4.34 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Produk.



Gambar 4.34. Activity Diagram Melihat Produk

F-031. Menambah Produk

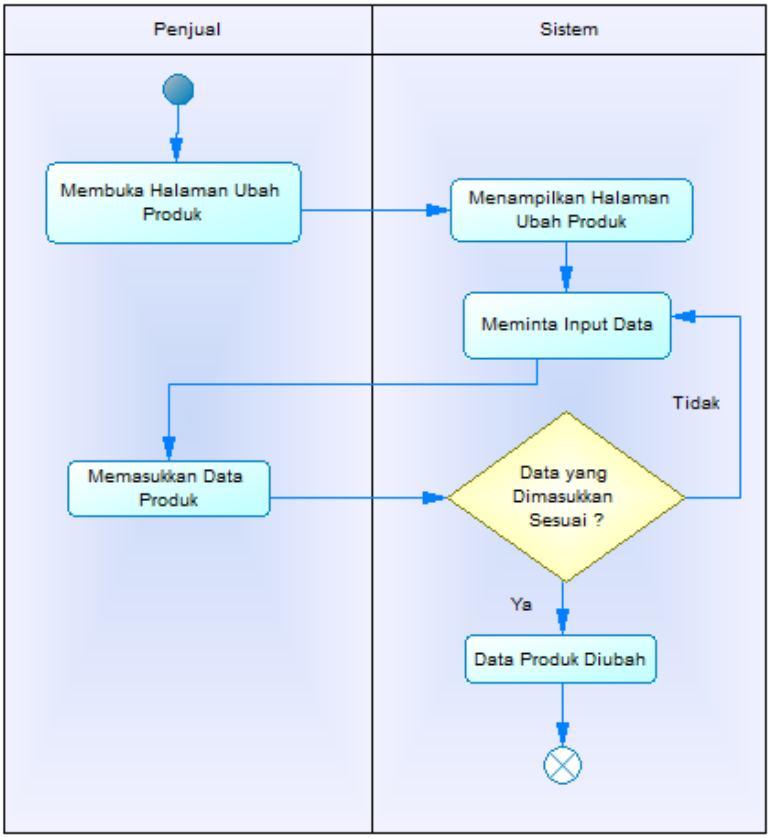
Penjual dapat Menambah Produk. Gambar 4.35 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Menambah Produk.



Gambar 4.35. Activity Diagram Menambah Produk

F-032. Mengubah Produk

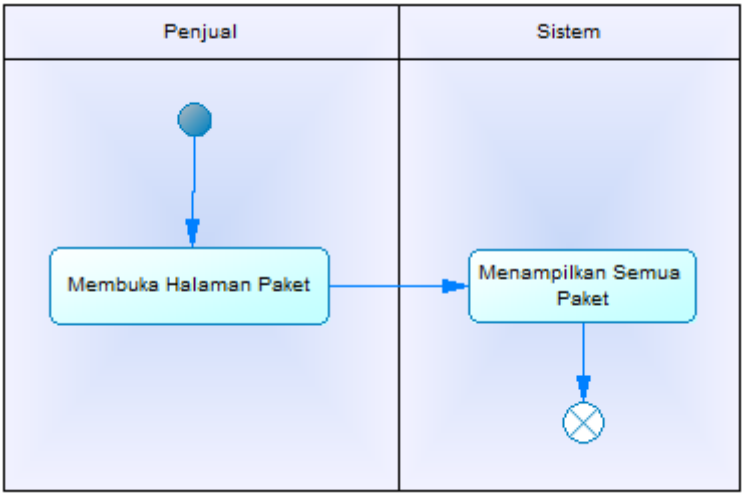
Penjual dapat Mengubah Produk. Gambar 4.36 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Mengubah Produk.



Gambar 4.36. Activity Diagram Mengubah Produk

F-033. Melihat Paket

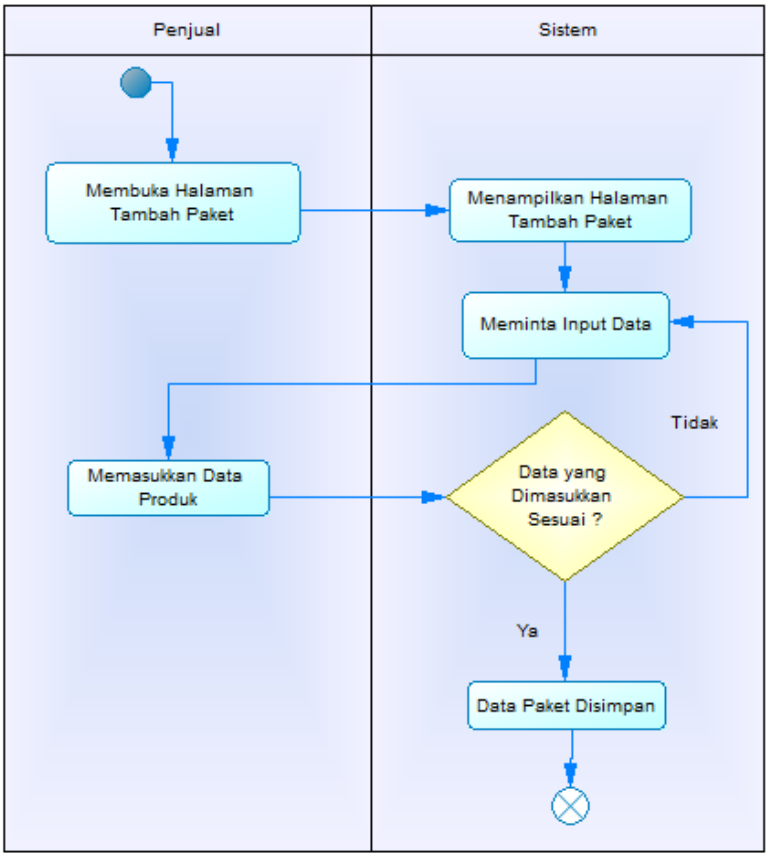
Penjual dapat Melihat Paketnya. Gambar 4.37 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Paket.



Gambar 4.37. Activity Diagram Melihat Paket

F-034. Menambah Paket

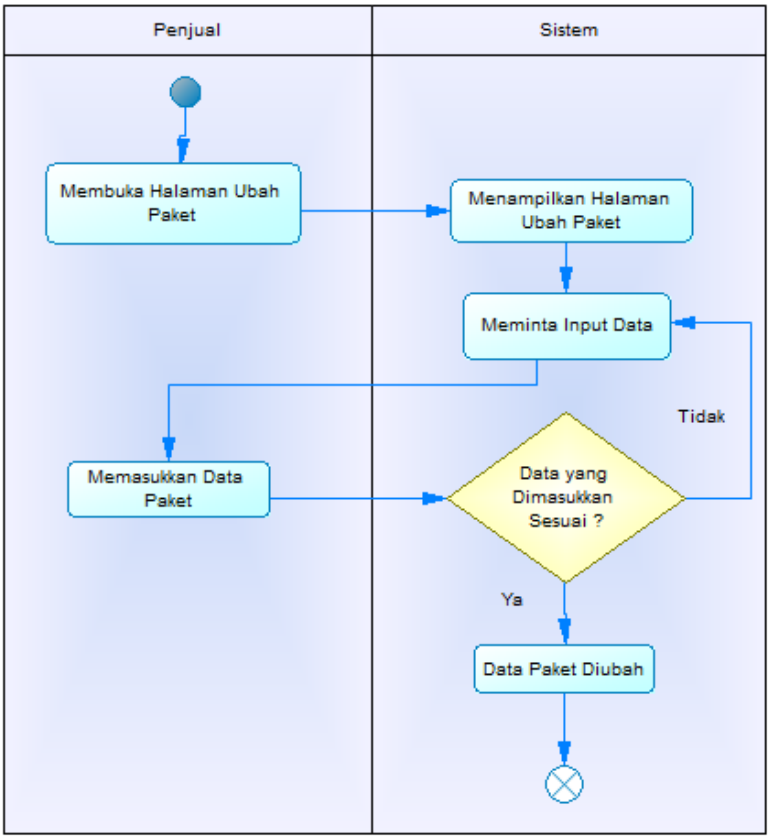
Penjual dapat Menambah Paket. Gambar 4.38 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Menambah Paket.



Gambar 4.38. Activity Diagram Menambah Paket

F-035. Mengubah Paket

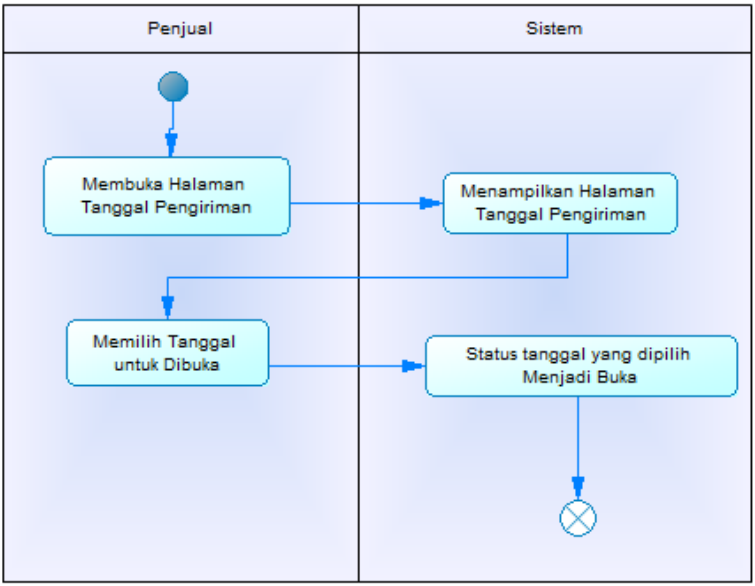
Penjual dapat Mengubah Paket. Gambar 4.39 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Mengubah Paket.



Gambar 4.39. Activity Diagram Mengubah Paket

F-036. Membuka Tanggal Pengiriman

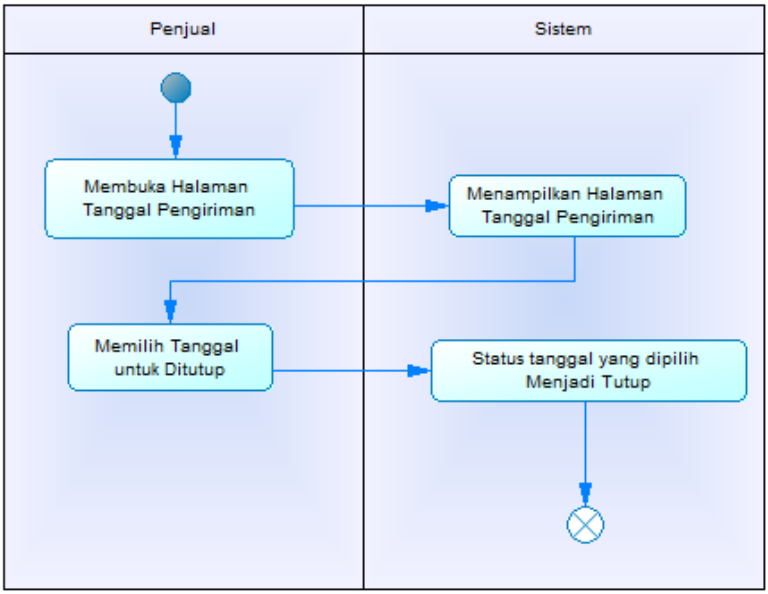
Penjual dapat Membuka Tanggal Pengiriman. Gambar 4.40 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Membuka Tanggal Pengiriman.



Gambar 4.40. *Activity Diagram* Membuka Tanggal Pengiriman

F-037. Menutup Tanggal Pengiriman

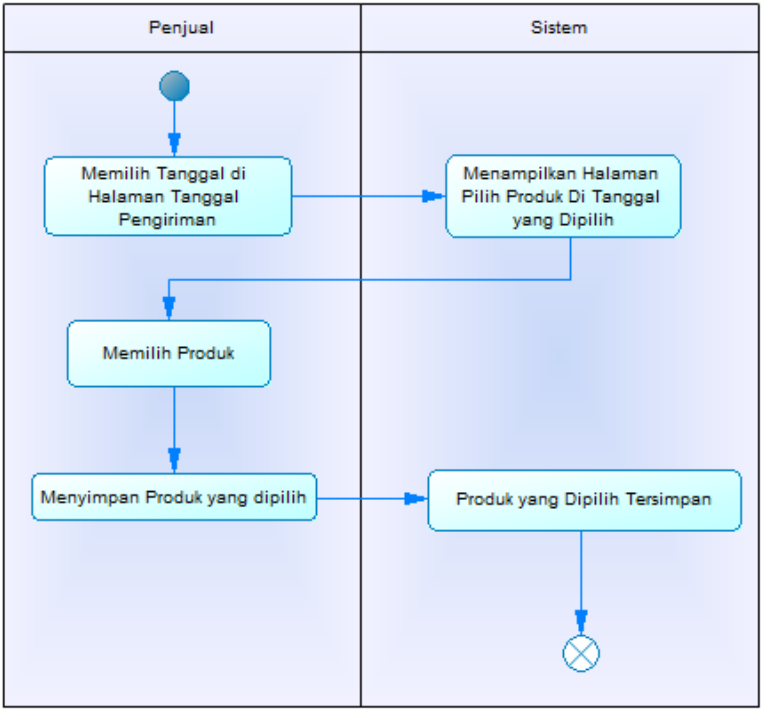
Penjual dapat Menutup Tanggal Pengiriman yang tersedia. Gambar 4.41 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Menutup Tanggal Pengiriman.



Gambar 4.41. Activity Diagram Menutup Tanggal Pengiriman

F-038. Memilih Produk Pada Tanggal Pengiriman

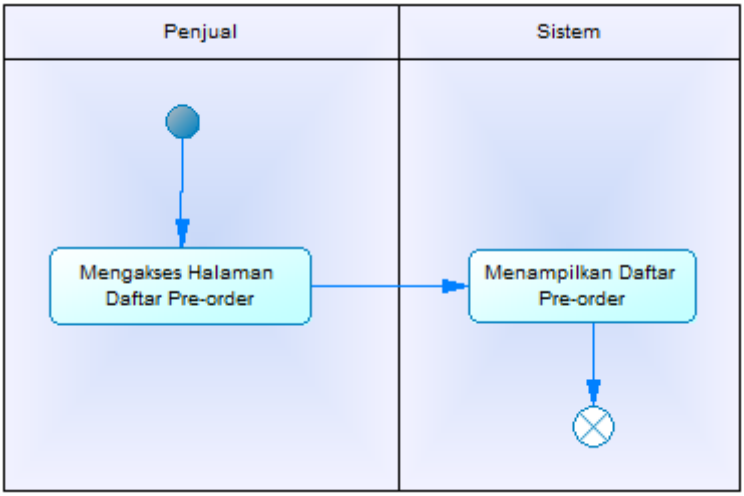
Penjual dapat Memilih Produk Pada Tanggal Pengiriman yang tersedia. Gambar 4.42 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Memilih Produk Pada Tanggal Pengiriman.



Gambar 4.42. Activity Diagram Memilih Produk Pada Tanggal Pengiriman

F-039. Melihat Daftar *Pre-order* Pembeli

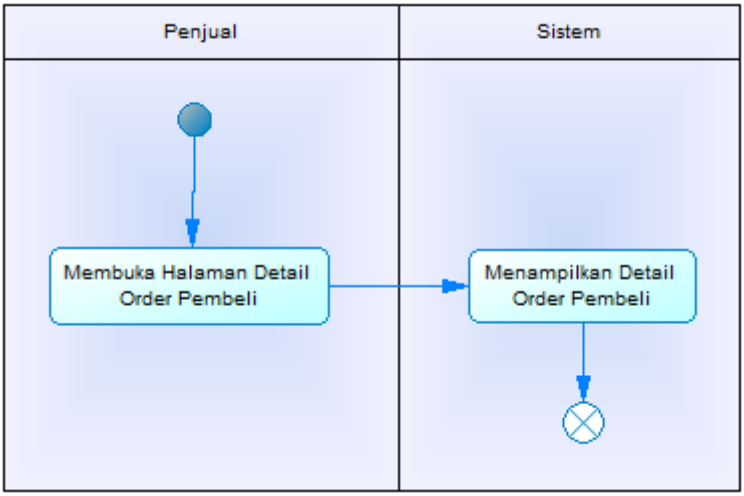
Penjual dapat melihat daftar *Pre-order* Pembeli di tanggal tertentu. Gambar 4.43 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Daftar *Pre-order* Pembeli.



Gambar 4.43. Activity Diagram Melihat Daftar Pre-order Pembeli

F-040. Melihat Detail *Order* Pembeli

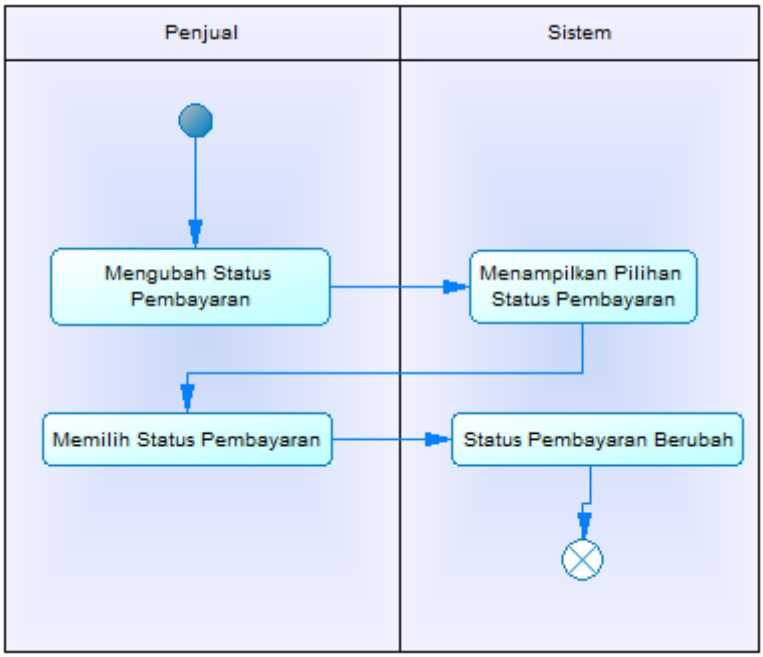
Penjual dapat melihat detail *order* Pembeli di. Gambar 4.44 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Detail *Order* Pembeli.



Gambar 4.44. Activity Diagram Melihat Detail Order Pembeli

F-041. Mengubah Status Pembayaran Pembeli

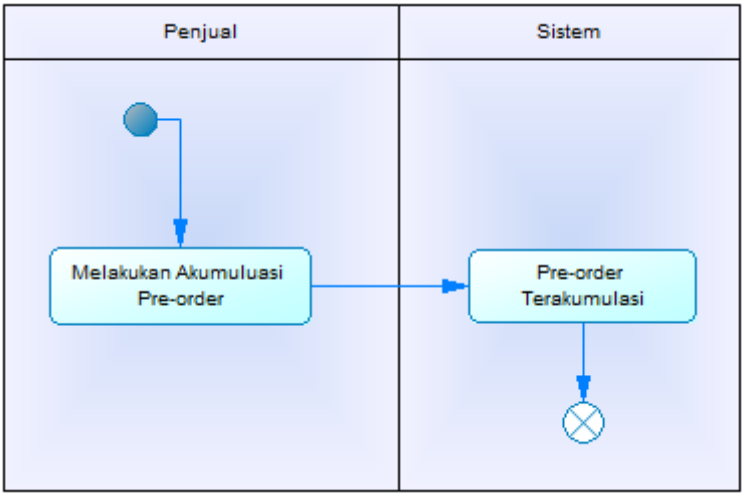
Penjual dapat mengubah status pembayaran Pembeli (Lunas, Belum Dibayar, Gagal) di. Gambar 4.45 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Mengubah Status Pembayaran Pembeli.



Gambar 4.45. *Activity Diagram* Mengubah Status Pembayaran Pembeli

F-042. Melakukan Akumulasi *Pre-order*

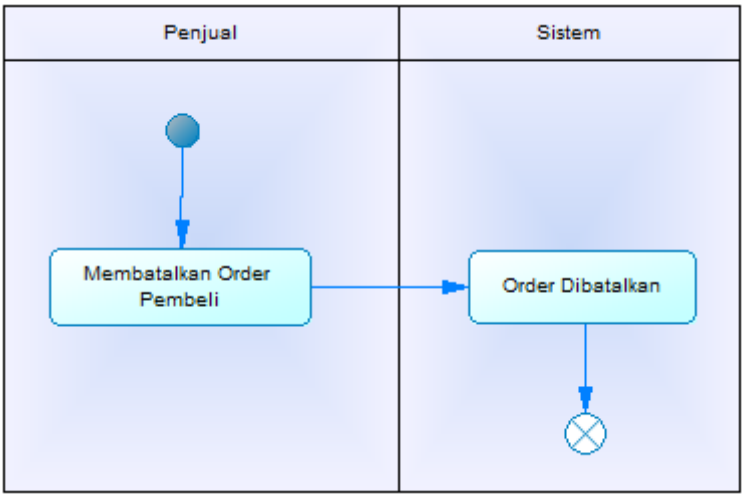
Penjual dapat melakukan akumulasi *Pre-order* di. Gambar 4.46 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melakukan Akumulasi *Pre-order*.



Gambar 4.46. Activity Diagram Melakukan Akumulasi Pre-order

F-043. Membatalkan Order Pembeli

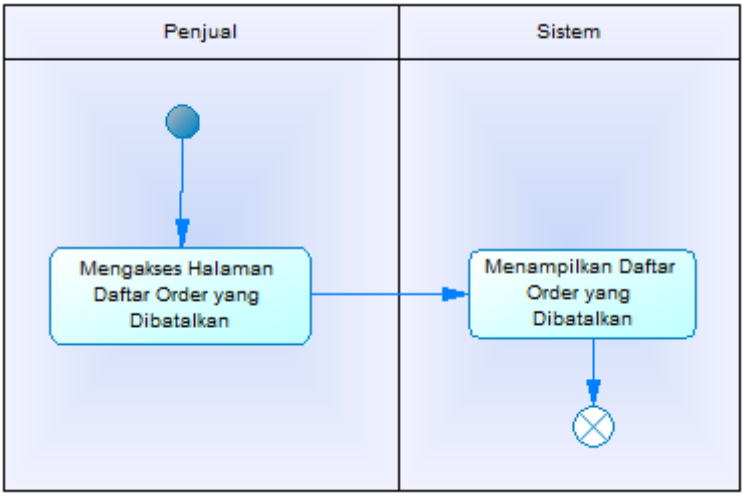
Penjual dapat membatalkan *order* Pembeli di. Gambar 4.47 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Membatalkan *Order* Pembeli.



Gambar 4.47. Activity Diagram Membatalkan Order Pembeli

F-044. Melihat Daftar *Order* yang Dibatalkan

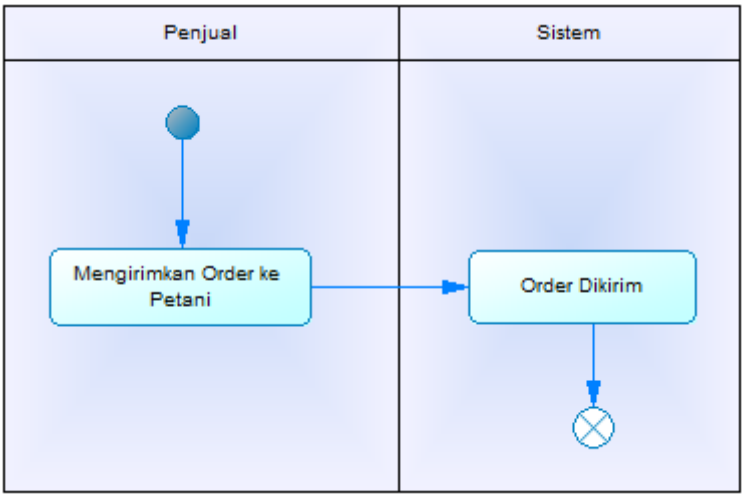
Penjual dapat melihat daftar *order* yang dibatalkan di. Gambar 4.48 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Daftar *Order* yang Dibatalkan.



Gambar 4.48. Activity DiagramMelihat Daftar Order yang Dibatalkan

F-045. Mengirimkan *Order* ke Petani

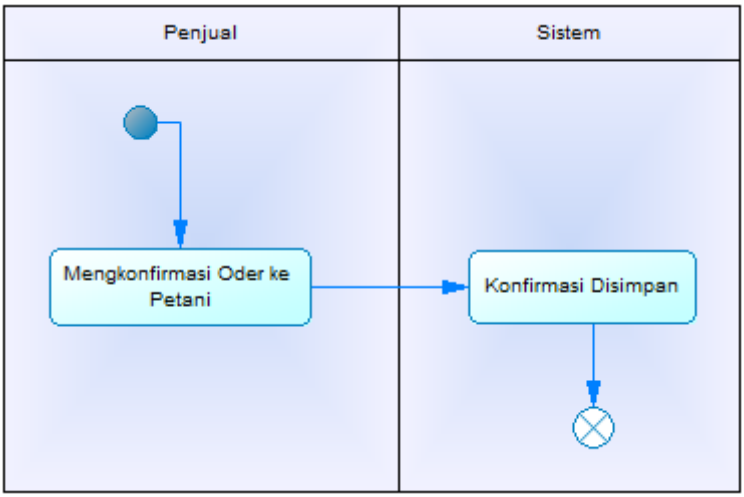
Penjual dapat mengirimkan *order* ke Petani di. Gambar 4.49 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Mengirimkan *Order* ke Petani.



Gambar 4.49. Activity Diagram Mengirimkan Order ke Petani

F-046. Melakukan Konfirmasi *Order* ke Petani

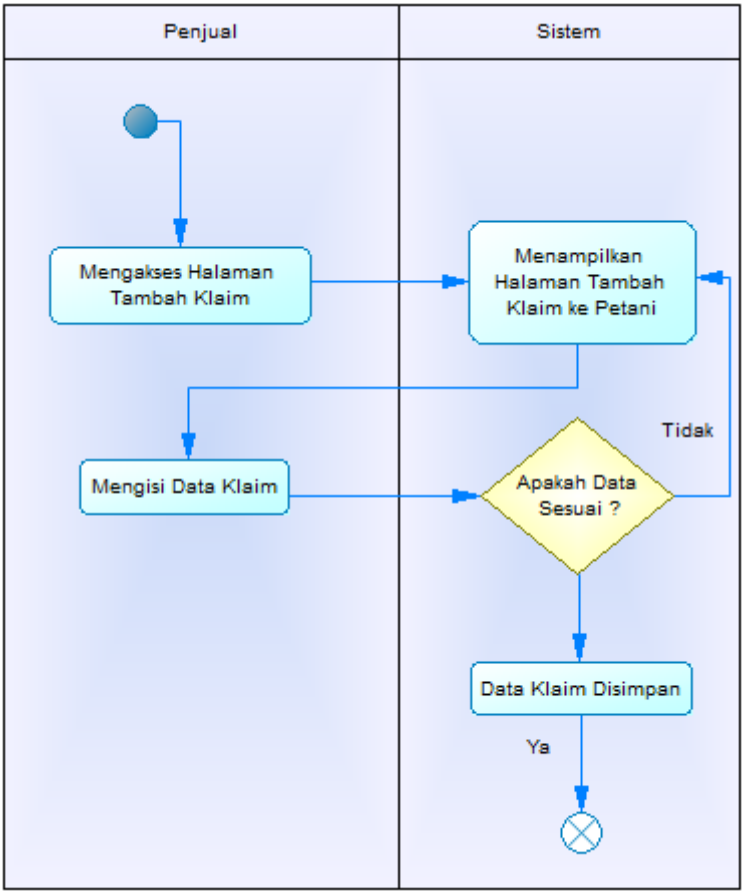
Penjual dapat melakukan konfirmasi *order* ke Petani di. Gambar 4.50 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melakukan Konfirmasi *Order* ke Petani.



Gambar 4.50. Activity Diagram Melakukan Konfirmasi Order ke Petani

F-047. Melakukan Klaim *Order* ke Petani

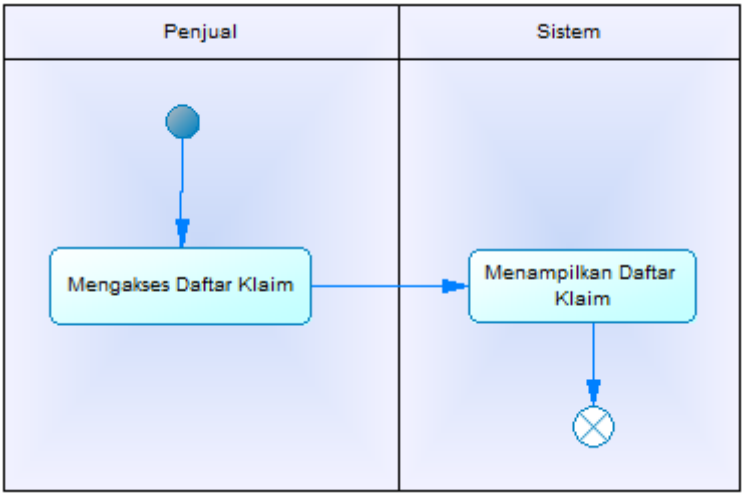
Penjual dapat melakukan klaim *order* ke Petani di. Gambar 4.51 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melakukan Klaim *Order* ke Petani.



Gambar 4.51. Activity Diagram Melakukan Klaim Order ke Petani

F-048. Melihat Daftar Klaim

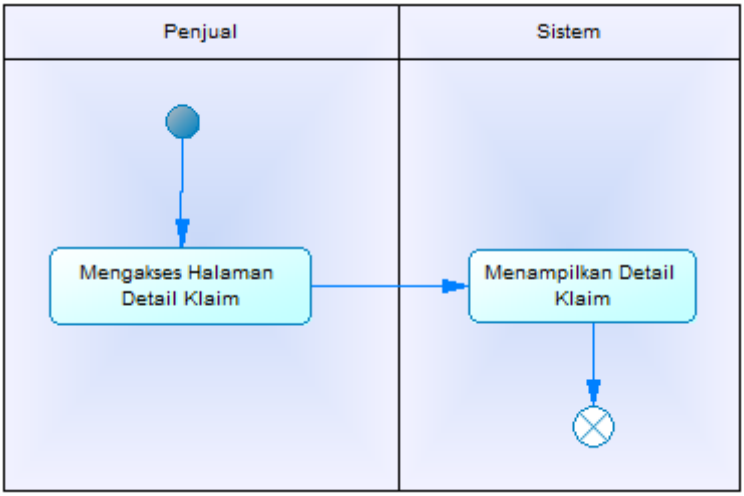
Penjual dapat melihat data klaim order ke Petani di. Gambar 4.52 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Daftar Klaim.



Gambar 4.52. Activity Diagram Melihat Daftar Klaim

F-049. Melihat Detail Klaim

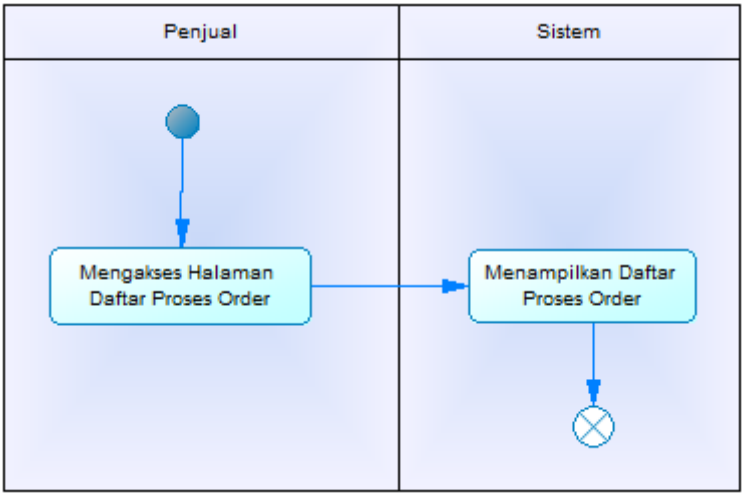
Penjual dapat melihat detail klaim *order* ke Petani di. Gambar 4.53 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Detail Klaim.



Gambar 4.53. Activity Diagram Melihat Detail Klaim

F-050. Melihat Daftar Proses *Order* Pembeli

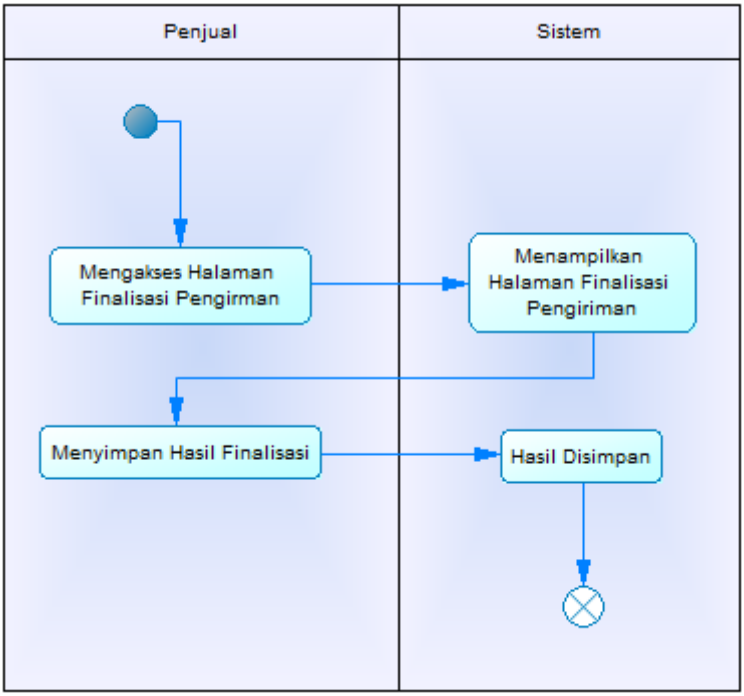
Penjual dapat melihat daftar proses *order* Pembeli di. Gambar 4.54 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Daftar Proses *Order* Pembeli.



Gambar 4.54. Activity Diagram Melihat Daftar Proses Order Pembeli

F-051. Melakukan Finalisasi Pengiriman ke Pembeli

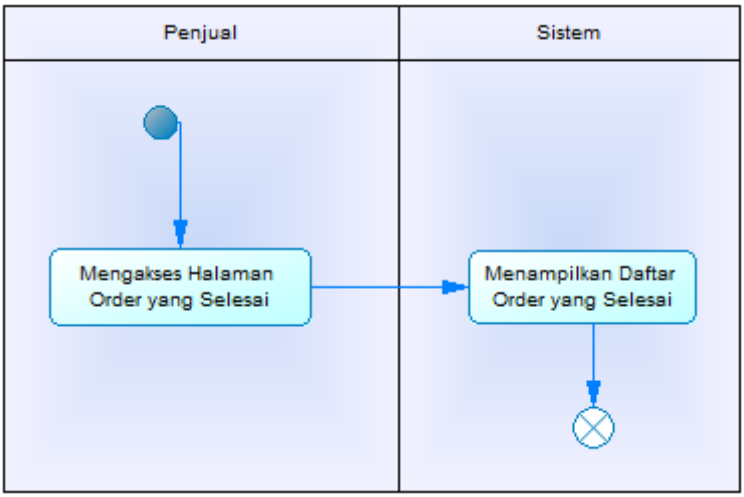
Penjual dapat melakukan finalisasi pengiriman ke Pembeli di. Gambar 4.55 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melakukan Finalisasi Pengiriman ke Pembeli.



Gambar 4.55. Activity Diagram Melakukan Finalisasi Pengiriman ke Pembeli

F-052. Melihat Daftar *Order* yang Selesai

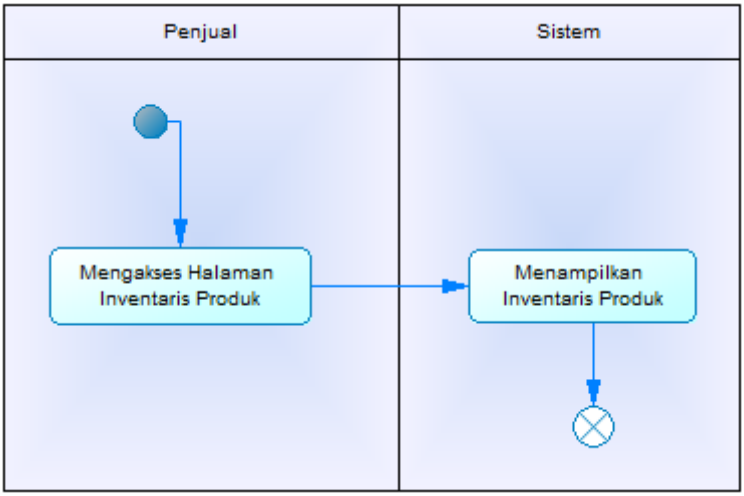
Penjual dapat melihat daftar *order* yang selesai di. Gambar 4.56 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Daftar *Order* yang Selesai.



Gambar 4.56. Activity Diagram Melihat Daftar Order yang Selesai

F-053. Melihat Inventaris Produk

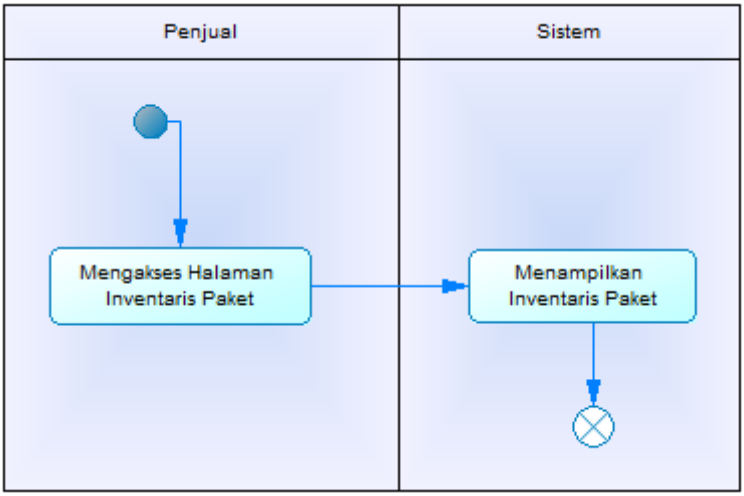
Penjual dapat melihat inventaris Produk di. Gambar 4.57 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Inventaris Produk.



Gambar 4.57. Activity Diagram Melihat Inventaris Produk

F-054. Melihat Inventaris Paket

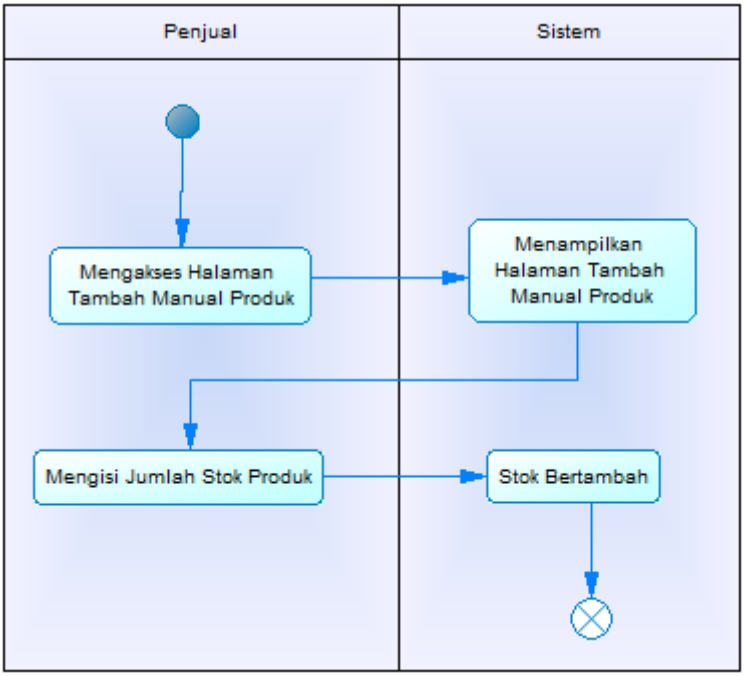
Penjual dapat melihat inventaris Paket di. Gambar 4.58 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Inventaris Paket.



Gambar 4.58. Activity Diagram Melihat Inventaris Paket

F-055. Menambah Manual Stok Produk

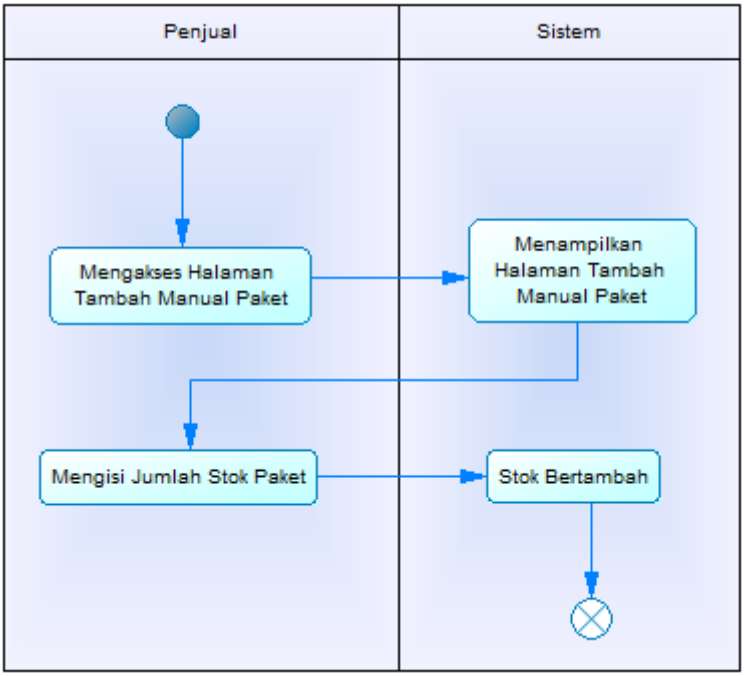
Penjual dapat Menambah manual stok Produk di. Gambar 4.59 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Menambah Manual Stok Produk.



Gambar 4.59. Activity Diagram Menambah Manual Stok Produk

F-056. Menambah Manual Stok Paket

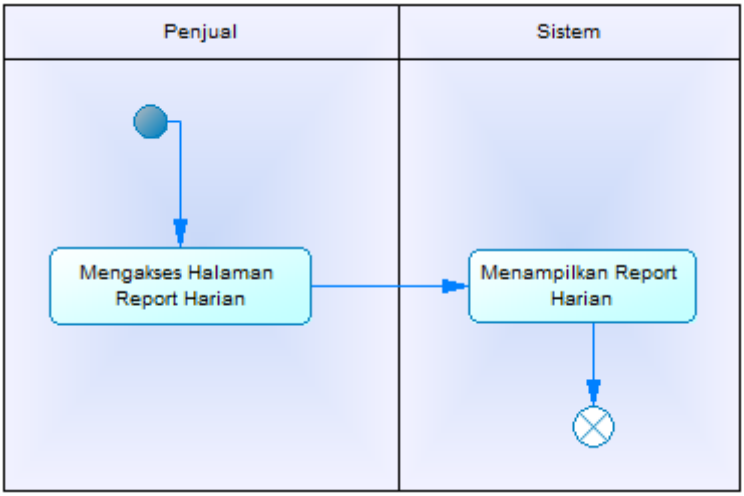
Penjual dapat Menambah manual stok Paket di. Gambar 4.60 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Menambah Manual Stok Paket.



Gambar 4.60. *Activity Diagram* Menambah Manual Stok Paket

F-057. Melihat *Report* Harian

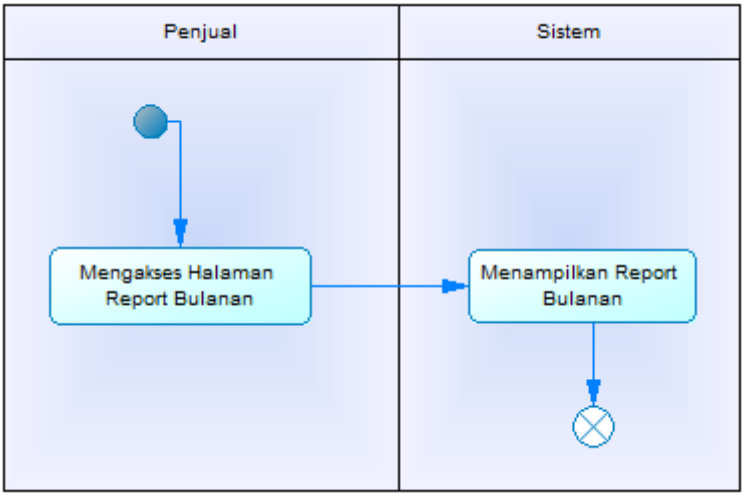
Penjual dapat melihat *report* harian di. Gambar 4.61 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat *Report* Harian.



Gambar 4.61. Activity Diagram Melihat Report Harian

F-058. Melihat *Report* Bulanan

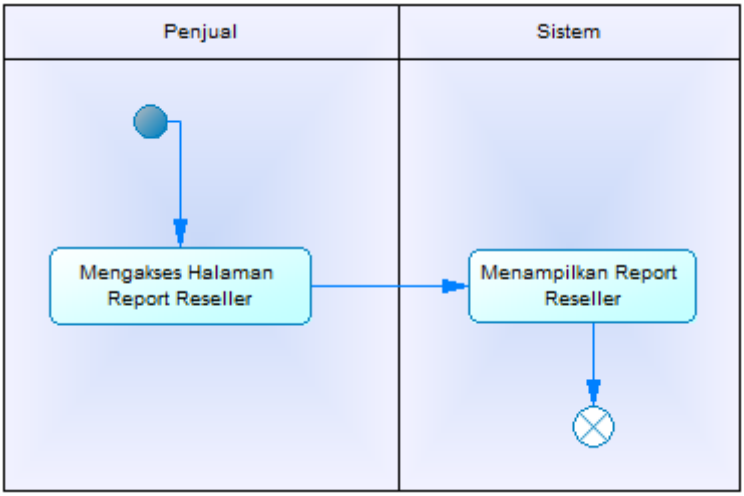
Penjual dapat melihat *report* bulanan di. Gambar 4.62 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat *Report* Bulanan.



Gambar 4.62. Activity Diagram Melihat Report Bulanan

F-059. Melihat *Report* Reseller

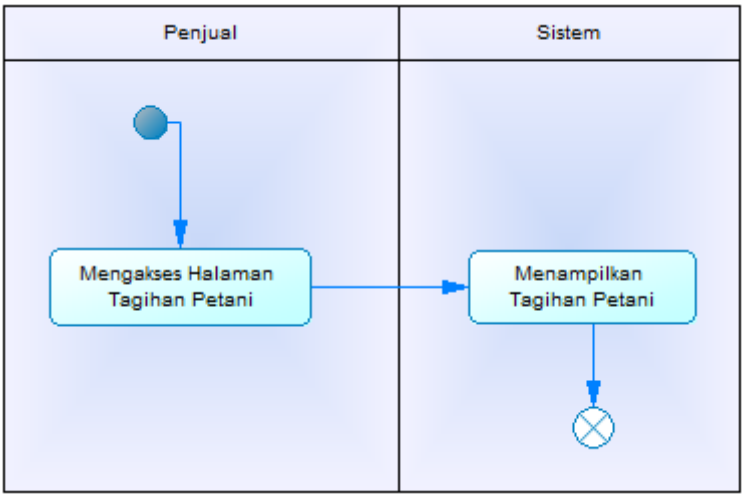
Penjual dapat melihat *report* Reseller di. Gambar 4.63 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat *Report* Reseller.



Gambar 4.63. Activity Diagram Melihat Report Reseller

F-060. Melihat Daftar Tagihan ke Petani

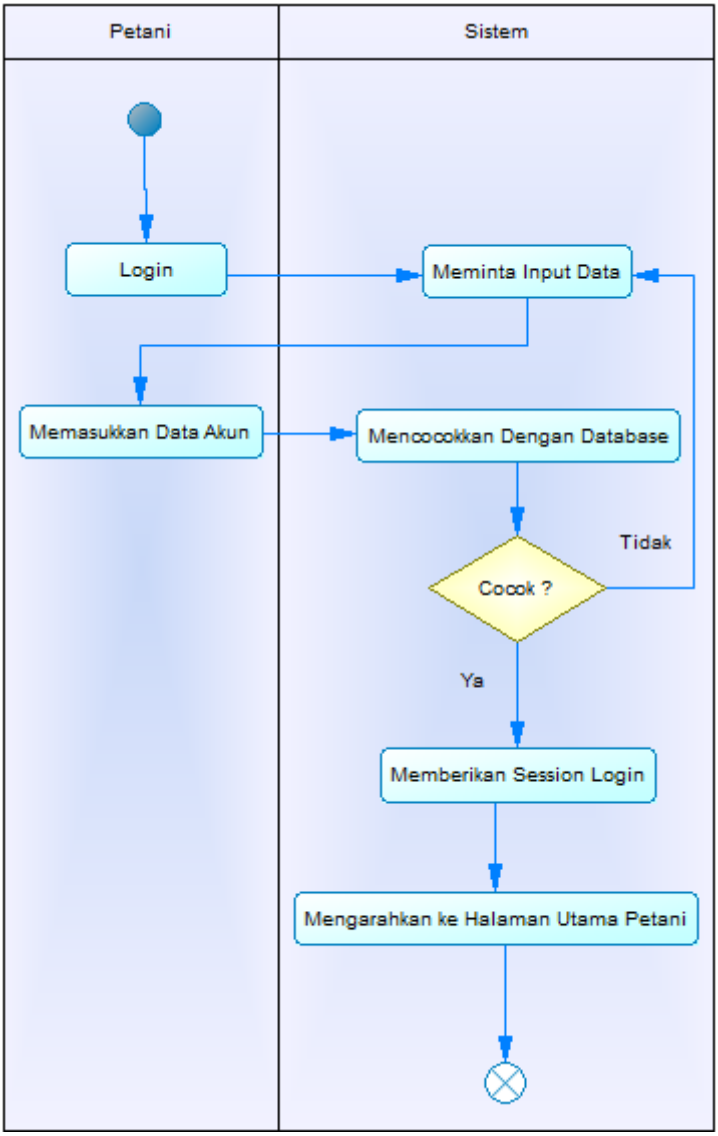
Penjual dapat melihat melihat daftar tagihan ke Petani di. Gambar 4.64 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Daftar Tagihan ke Petani.



Gambar 4.64. Activity Diagram Melihat Daftar Tagihan ke Petani

F-061. Autentikasi Petani

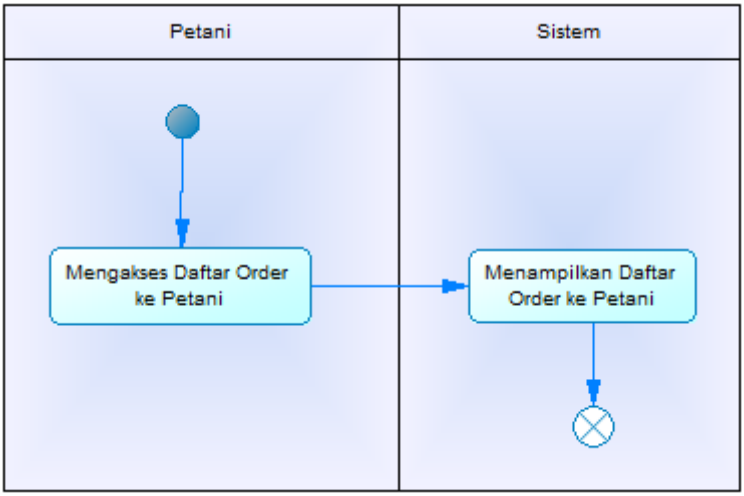
Petani melakukan *login* kedalam aplikasi untuk memastikan keaslian hak akses di. Gambar 4.65 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Autentikasi.



Gambar 4.65. Activity Diagram Autentikasi Petani

F-062. Melihat Daftar *Order* ke Petani

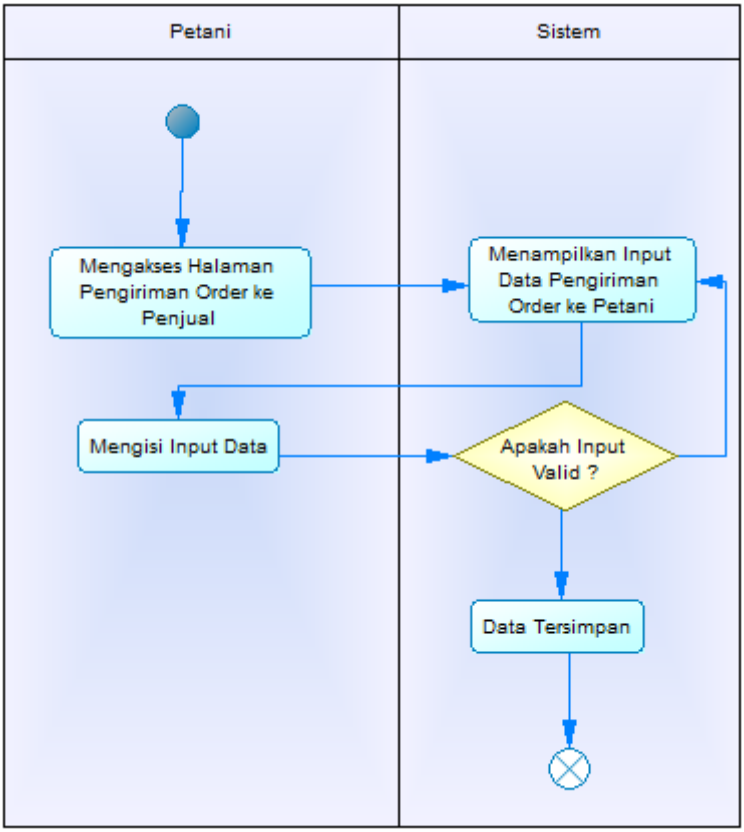
Petani melihat daftar order ke Petani di. Gambar 4.66 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Daftar *Order* ke Petani.



Gambar 4.66. Activity Diagram Melihat Daftar Order ke Petani

F-063. Melakukan Pengiriman *Order* ke Penjual

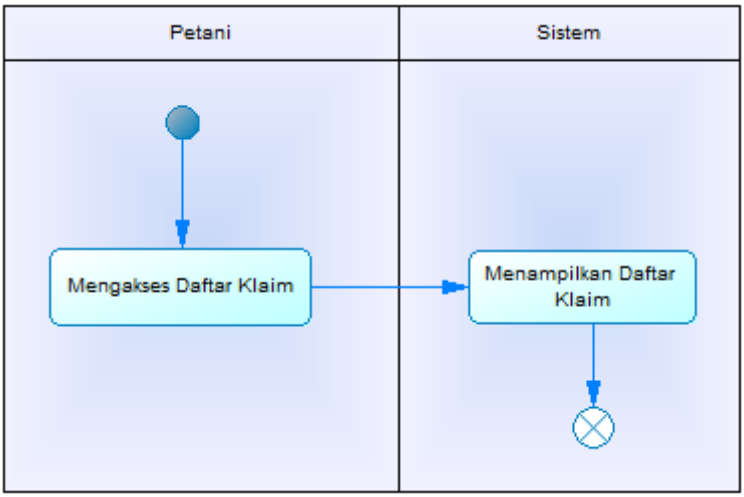
Petani melakukan pengiriman order ke Penjual di. Gambar 4.67 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melakukan Pengiriman *Order* ke Penjual.



Gambar 4.67. Activity Diagram Melakukan Pengiriman Order ke Penjual

F-064. Melihat Daftar Klaim

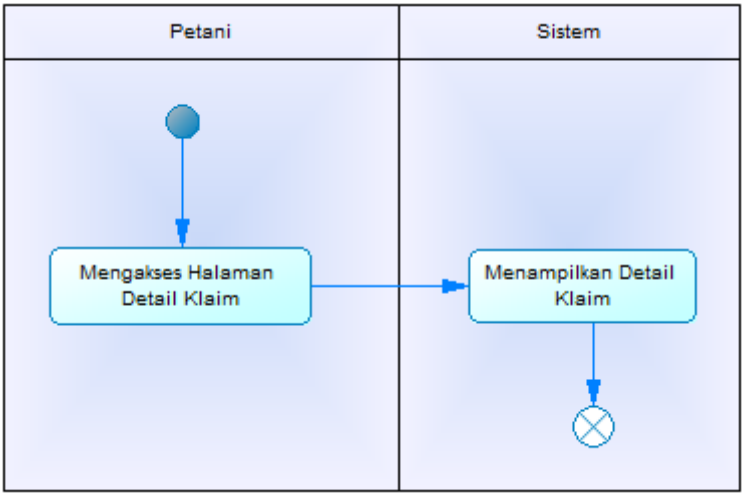
Petani melihat daftar klaim di. Gambar 4.68 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Daftar Klaim.



Gambar 4.68. Activity Diagram Melihat Daftar Klaim

F-065. Melihat Detail Klaim

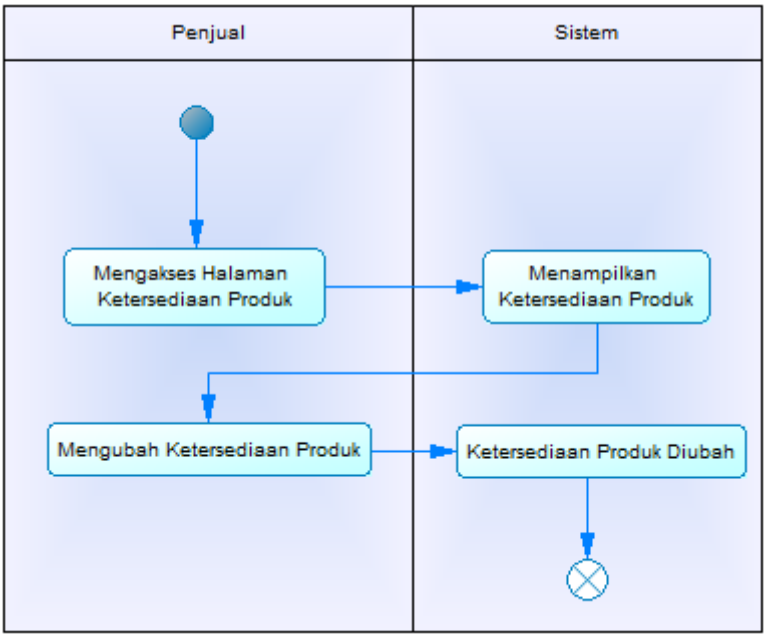
Petani melihat detail klaim di. Gambar 4.69 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Detail Klaim.



Gambar 4.69. Activity Diagram Melihat Detail Klaim

F-066. Mengatur Ketersediaan Produk yang Dimiliki

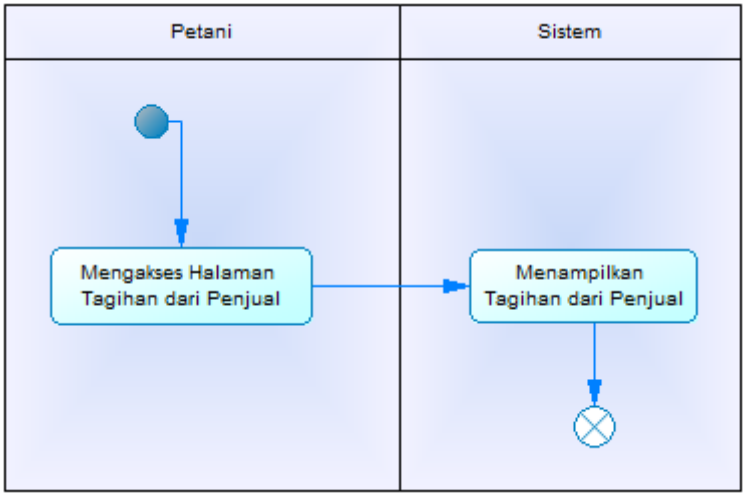
Petani mengatur ketersediaan produk yang dimilikinya di. Gambar 4.70 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Mengatur Ketersediaan Produk yang Dimilikinya.



Gambar 4.70. Activity Diagram Mengatur Ketersediaan Produk yang Dimilikinya

F-067. Melihat Tagihan dari Penjual

Petani melihat tagihan dari Penjual di. Gambar 4.71 di bawah ini merupakan diagram aktivitas yang menunjukkan alur untuk Melihat Tagihan dari Penjual.



Gambar 4.71. Activity Diagram Melihat Tagihan dari Penjual

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari sistem yang kami buat. Implementasi ini akan dibagi ke dalam beberapa bagian, yaitu bagian implementasi lapisan model, implementasi lapisan repository, implementasi lapisan kontrol, dan implementasi antarmuka pengguna.

5.1. Implementasi Model

Implementasi lapisan model ini merupakan lapisan yang digunakan *framework* Laravel untuk menghubungkan aplikasi dengan *database.* Adapun implementasi lapisan model sebagai berikut:

5.1.1. Lapisan Model Alamat

Bagian dari implementasi lapisan model Alamat yang menyimpan data Alamat Pembeli ditampilkan pada Kode Sumber 5.1 berikut:

```
1.  <?php
2.
3.  namespace App\Models;
4.
5.  use Illuminate\Database\Eloquent\Model;
6.
7.  class Alamat extends Model
8.  {
9.      protected $table = 'alamats';
10.
11.     protected $fillable = [
```

```
        'id', 'id_user', 'kotkab', 'daerah', 'kodepos'
    , 'long', 'lat',
        'blok_nomor', 'alamat', 'info_tambahan'
    ];

    public $incrementing = false;
}
```

Kode Sumber 5.1. Model Alamat

5.1.2. Lapisan Model Barang_tanggal

Bagian dari implementasi lapisan model Barang_tanggal yang menyimpan data barang apa yang tersedia di tanggal tertentu ditampilkan pada Kode Sumber 5.2 berikut:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use App\User;

class Barang_tanggal extends Model
{
    protected $fillable = [
        'id',
        'id_barang',
        'tanggal',
    ];

    public $incrementing = false;

    public function barang()
    {
```

```
20.           return $this->
      belongsTo('App\Models\Barang', 'id_barang', 'id');

21.       }
22.
23.   }
```

Kode Sumber 5.2. Model Barang_tanggal

5.1.3. Lapisan Model Barang

Bagian dari implementasi lapisan model Barang yang menyimpan data Barang ditampilkan pada Kode Sumber 5.3 berikut:

```
1.    <?php
2.
3.    namespace App\Models;
4.
5.    use Illuminate\Database\Eloquent\Model;
6.    use App\User;
7.
8.    class Barang extends Model
9.    {
10.       protected $table = 'barangs';
11.       protected $fillable = [
12.           'id',
13.           'id_user',
14.           'id_kategori',
15.           'nama',
16.           'kode',
17.           'jenis',
18.           'satuan',
19.           'bobot',
20.           'harga_beli',
21.           'harga_jual',
22.           'deskripsi',
23.           'diskon',
```

```
            'jenis_diskon',
            'show_etalase',
            'is_paket',
            'ketersediaan',
            'stok',
            'bobot_minimum_timbang',
            'bobot_kemasan_kemas',
            'jenis_diskon_reseller',
            'diskon_reseller',
            'harga_jual_reseller',
            'jumlah_pcs'
        ];

        public $incrementing = false;

        public function supplier(){
            return $this->hasOne('App\User','id','id_user');
        }

        public function isiPaket(){
            return $this->hasMany('App\Models\Isi_paket','id_barang_parent','id');
        }

        public function detailTransaksi(){
            return $this->hasMany('App\Models\Detail_Transaksi','id_barang','id');
        }

        public function kategori(){
            return $this->hasOne('App\Models\Base_Kategori','id','id_kategori');
        }
```

```
54.
55.         public function foto_barang(){
56.             return $this-
        >hasMany('App\Models\Foto_barang','id_barang','id')
        ;
57.         }
58.
59.         public function barangTanggal()
60.         {
61.             return $this-
        >hasMany('App\Models\Barang_tanggal', 'id_barang',
        'id');
62.         }
63.
64.     }
```

Kode Sumber 5.3. Model Barang

5.1.4. Lapisan Model Barangs_group

Bagian dari implementasi lapisan model Barang_group yang menyimpan data barang beserta kategorinya ditampilkan pada Kode Sumber 5.4 berikut:

```
1.      <?php
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Barangs_group extends Model
8.      {
9.          //
10.
11.         protected $fillable = [
12.             'id',
```

```
            'id_barang',
            'id_kategori'
        ];


        public $incrementing = false;
    }
```

Kode Sumber 5.4. Model Barangs_tanggal

5.1.5. Lapisan Model Barangs_kemasan

Bagian dari implementasi lapisan model Barang_kemasan yang menyimpan data barang beserta kemasannya ditampilkan pada Kode Sumber 5.5 berikut:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Barangs_kemasan extends Model
{
    //

    protected $fillable = [
        'id',
        'id_barang',
        'bobot_kemasan',
    ];

    public $incrementing = false;
}
```

Kode Sumber 5.5. Model Barangs_kemasan

5.1.6. Lapisan Model Base_Kategori

Bagian dari implementasi lapisan model Base_Kategori yang menyimpan data Kategori barang ditampilkan pada Kode Sumber 5.6 berikut:

```
1.  <?php
2.
3.  namespace App\Models;
4.
5.  use Illuminate\Database\Eloquent\Model;
6.
7.  class Base_Kategori extends Model
8.  {
9.      protected $table = 'base_kategoris';
10.
11.     protected $fillable = [
12.         'kategori'
13.     ];
14.
15.     public $incrementing = false;
16. }
```

Kode Sumber 5.6. Model Base_Kategori

5.1.7. Lapisan Model Bobot_Kemasan

Bagian dari implementasi lapisan model Bobot_Kemasan yang menyimpan data Bobot Kemasan dari barangnya ditampilkan pada Kode Sumber 5.7 berikut:

```
1.  <?php
2.
3.  namespace App\Models;
4.
5.  use Illuminate\Database\Eloquent\Model;
```

```
6.
7.      class Bobot_Kemasan extends Model
8.      {
9.          protected $table = 'bobot_kemasans';
10.
11.         protected $fillable = [
12.             'bobot_kemasan'
13.         ];
14.
15.         public $incrementing = false;
16.
17.     }
```

Kode Sumber 5.7. Model Bobot_Kemasan

5.1.8. Lapisan Model Detail_keranjang

Bagian dari implementasi lapisan model Detail_keranjang yang menyimpan data Detail dari sebuah Keranjang ditampilkan pada Kode Sumber 5.8 berikut:

```
1.      <?php
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Detail_keranjang extends Model
8.      {
9.          protected $table = 'detail_keranjangs';
10.
11.         protected $fillable = [
12.             'id_keranjang', 'id_barang', 'volume', 'har
        ga', 'id', 'bobot_kemasan', 'harga_diskon'
13.         ];
14.
```

```
15.         public $incrementing = false;
16.     }
```

Kode Sumber 5.8. Model Detail_keranjang

5.1.9. Lapisan Model Detail_klaim

Bagian dari implementasi lapisan model Detail_klaim yang menyimpan data Detail dari sebuah Klaim ditampilkan pada Kode Sumber 5.9 berikut:

```
1.      <?php
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Detail_klaim extends Model
8.      {
9.          //
10.         public $fillable = [
11.             'id',
12.             'id_klaim',
13.             'id_barang',
14.             'volume_klaim',
15.             'keterangan',
16.             'foto_bukti',
17.             'id_detail_transaksi'
18.         ];
19.         public $incrementing = false;
20.
21.         public function barang(){
22.             return $this-
        >hasOne('App\Models\Barang','id','id_barang');
23.         }
24.         public function klaim(){
25.             return $this-
        >hasOne('App\Models\Klaim','id','id_klaim');
```

```
26.         }
27.         public function detail_transaksi(){
28.             return $this-
        >hasOne('App\Models\Detail_transaksi','id','id_deta
        il_transaksi');
29.         }
30.     }
```

Kode Sumber 5.9. Model Detail_klaim

5.1.10. Lapisan Model Detail_transaksi

Bagian dari implementasi lapisan model Detail_transaksi yang menyimpan data Detail dari sebuah Transaksi ditampilkan pada Kode Sumber 5.10 berikut:

```
1.      <?php
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Detail_transaksi extends Model
8.      {
9.          protected $table = 'detail_transaksis';
10.
11.         protected $fillable = [
12.             'bobot_kemasan',
13.             'id_transaksi',
14.             'id_barang',
15.             'volume',
16.             'harga',
17.             'harga_diskon',
18.             'volume_terima',
19.             'volume_selisih',
20.             'volume_kirim_petani',
```

```
            'volume_kirim_kurir',
            'bobot_kirim_kurir',
            'harga_akhir',
            'keterangan',
            'id',
            'id_keranjang',
            'id_barang',
            'volume',
            'harga_akhir',
            'harga_akhir_diskon',
            'is_info_petani',
            'is_canceled_by_veggo',
            'is_include_rekap',
        ];

        public $incrementing = false;

        public function barang(){
            return $this->hasOne('App\Models\Barang','id','id_barang');
        }

        public function transaksi(){
            return $this->hasOne('App\Models\Transaksi','id','id_transaksi');
        }
    }
```

Kode Sumber 5.10. Model Detail_transaksi

5.1.11. Lapisan Model Foto_barang

Bagian dari implementasi lapisan model Foto_barang yang menyimpan data Foto dari barang ditampilkan pada Kode Sumber 5.11 berikut:

```
<?php
```

```
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Foto_barang extends Model
8.      {
9.          //
10.         protected $fillable = [
11.             'id',
12.             'id_barang',
13.             'path',
14.         ];
15.
16.         public $incrementing = false;
17.
18.         public function barang(){
19.             return $this-
        >hasOne('App\Models\Barang','id','id_barang');
20.         }
21.     }
```

Kode Sumber 5.11. Model Foto_barang

5.1.12. Lapisan Model Inventaris

Bagian dari implementasi lapisan model Inventaris yang menyimpan data Inventaris ditampilkan pada Kode Sumber 5.12 berikut:

```
22.     <?php
23.
24.     namespace App\Models;
25.
26.     use Illuminate\Database\Eloquent\Model;
27.
28.     class Inventaris extends Model
29.     {
```

```
30.         protected $table = 'inventaris';
31.
32.         protected $fillable = [
33.             'id',
34.             'id_barang',
35.             'id_transaksi',
36.             'tanggal',
37.             'status',
38.             'keterangan',
39.             'jumlah',
40.         ];
41.
42.         public $incrementing = false;
43.
44.         public function barang(){
45.             return $this-
      >hasOne('App\Models\Barang','id','id_barang');
46.         }
47.
48.         public function transaksi(){
49.             return $this-
      >hasOne('App\Models\Transaksi','id','id_transaksi')
      ;
50.         }
51.     }
```

Kode Sumber 5.12. Model Inventaris

5.1.13. Lapisan Model Isi_paket

Bagian dari implementasi lapisan model Isi_paket yang menyimpan data Isi dari sebuah Paket ditampilkan pada Kode Sumber 5.13 berikut:

```
1.  <?php
2.
3.  namespace App\Models;
```

```
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Isi_paket extends Model
8.      {
9.          protected $table = 'isi_pakets';
10.
11.         protected $fillable = [
12.             'id',
13.             'id_barang_parent',
14.             'id_barang',
15.             'volume',
16.             'harga',
17.         ];
18.
19.         public $incrementing = false;
20.
21.         public function barang(){
22.             return $this-
        >hasOne('App\Models\Barang','id','id_barang');
23.         }
24.     }
```

Kode Sumber 5.13. Model Isi_paket

5.1.14. Lapisan Model Kategori

Bagian dari implementasi lapisan model Kategori yang menyimpan data Kategori dari barang ditampilkan pada Kode Sumber 5.14 berikut:

```
1.      <?php
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
```

```
7.      class Kategori extends Model
8.      {
9.          protected $table = 'kategoris';
10.
11.         protected $fillable = [
12.             'id',
13.             'kategori',
14.             'sub_kategori',
15.         ];
16.
17.         public $incrementing = false;
18.
19.         public function baseKategori(){
20.             return $this-
        >hasOne('App\Models\Base_Kategori','id','kategori')
        ;
21.         }
22.     }
```

Kode Sumber 5.14. Model Kategori

5.1.15. Lapisan Model Keranjang

Bagian dari implementasi lapisan model Keranjang yang menyimpan data Keranjang ditampilkan pada Kode Sumber 5.15 berikut:

```
1.      <?php
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Keranjang extends Model
8.      {
9.          protected $table = 'keranjangs';
10.
```

```
11.         protected $fillable = [
12.             'id', 'id_user', 'tanggal_pre_order'
13.         ];
14.
15.         public $incrementing = false;
16.     }
```

Kode Sumber 5.15. Model Keranjang

5.1.16. Lapisan Model Klaim

Bagian dari implementasi lapisan model Klaim yang menyimpan data Klaim Penjual ke Petani ditampilkan pada Kode Sumber 5.16 berikut:

```
1.      <?php
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Klaim extends Model
8.      {
9.          //
10.
11.         public $fillable = [
12.             'id',
13.             'kode_klaim',
14.             'id_transaksi',
15.             'klaim_from',
16.             'klaim_to',
17.             'tanggal_kirim',
18.             'status',
19.         ];
20.
21.         public $incrementing = false;
22.
```

```
23.         public function dari(){
24.             return $this-
        >hasOne('App\User','id','klaim_from');
25.         }
26.
27.         public function untuk(){
28.             return $this-
        >hasOne('App\User','id','klaim_to');
29.         }
30.
31.         public function transaksi(){
32.             return $this-
        >hasOne('App\Models\Transaksi','id','id_transaksi')
        ;
33.         }
34.     }
```

Kode Sumber 5.16. Model Klaim

5.1.17. Lapisan Model Tanggal

Bagian dari implementasi lapisan model Tanggal yang menyimpan data Tanggal Buka Toko ditampilkan pada Kode Sumber 5.17 berikut:

```
1.      <?php
2.
3.      namespace App\Models;
4.
5.      use Illuminate\Database\Eloquent\Model;
6.
7.      class Tanggal extends Model
8.      {
9.          //
10.
11.         public $fillable = [
12.             'id',
```

```
            'tanggal',
            'flag',
        ];

        public $incrementing = false;
    }
```

Kode Sumber 5.17. Model Tanggal

5.1.18. Lapisan Model Transaksi

Bagian dari implementasi lapisan model Transaksi yang menyimpan data Transaksi ditampilkan pada Kode Sumber 5.18 berikut:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Transaksi extends Model
{
    protected $table = 'transaksis';

    protected $fillable = [
        'id',
        'id_user',
        'id_alamat',
        'id_kurir',
        'id_reseller',
        'nomor_invoice',
        'total_bayar',
        'status',
        'is_info_petani',
        'is_canceled_by_veggo',
        'bukti_transfer',
```

```
23.             'tanggal_pre_order',
24.             'keterangan',
25.             'isAlreadyPay',
26.             'tipe_transaksi',
27.             'tanggal_pengiriman',
28.             'tanggal_terima',
29.             'is_exclude_rekap',
30.             'is_confirm_finish_byuser',
31.             'is_diterima_reseller',
32.             'total_bayar_akhir',
33.             'ongkir'
34.         ];
35.
36.         public $incrementing = false;
37.
38.         public function user(){
39.             return $this-
        >hasOne('App\User','id','id_user');
40.         }
41.
42.         public function alamat(){
43.             return $this-
        >hasOne('App\Models\Alamat','id','id_alamat');
44.         }
45.
46.         public function kurir(){
47.             return $this-
        >hasOne('App\User','id','id_kurir');
48.         }
49.
50.         public function detailTransaksi(){
51.             return $this-
        >hasMany('App\Models\Detail_transaksi','id_transaks
        i','id');
52.         }
53.     }
```

Kode Sumber 5.18. Model Transaksi

5.1.19. Lapisan Model User

Bagian dari implementasi lapisan model User yang menyimpan data User ditampilkan pada Kode Sumber 5.19 berikut:

```
<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'id','name', 'email', 'password','nomor_hp', 'role', 'nomor_rek', 'bank', 'atas_nama'
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
```

```
28.             'password', 'remember_token',
29.         ];
30.
31.         /**
32.          * The attributes that should be cast to native
    types.
33.          *
34.          * @var array
35.          */
36.         protected $casts = [
37.             'email_verified_at' => 'datetime',
38.         ];
39.
40.         public $incrementing = false;
41.
42.     }
```

Kode Sumber 5.19. Model User

5.2. Implementasi Lapisan Repository

Implementasi lapisan repository ini merupakan lapisan yang digunakan sebagai lapisan pengakses *Model*. Adapun implementasi lapisan Repository sebagai berikut:

5.2.1. Lapisan Repository AlamatRepository

Lapisan Repository AlamatRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Alamat. Kode Sumber 5.20 berikut merupakan implementasi dari lapisan repository AlamatRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.
```

```
5.  use App\Models\Alamat;
6.  use GuzzleHttp\Client;
7.  use Webpatser\Uuid\Uuid;
8.  use App\Adapters\GeocodeAdapter;
9.  use Illuminate\Support\Facades\Auth;
10. use GuzzleHttp\Exception\GuzzleException;
11. use App\Repositories\Interfaces\AlamatInterface;
12.
13. class AlamatRepository implements AlamatInterface
14. {
15.     protected $model;
16.     protected $geocodeAdapter;
17.
18.     public function __construct(Alamat $model, Geocode
    Adapter $geocodeAdapter)
19.     {
20.         $this->model          = $model;
21.         $this->geocodeAdapter = $geocodeAdapter;
22.     }
23.
24.     public function getAllAlamatByUser($id)
25.     {
26.         return $this->model->where('id_user', $id)-
    >get();
27.     }
28.
29.     public function getAlamatById($id)
30.     {
31.         return $this->model->where('id', $id)-
    >get();
32.     }
33.
34.     public function hapusAlamat($id)
35.     {
36.         return $this->model->where('id', $id)-
    >delete();
37.     }
```

```
38.
39.     public function ubahAlamat($id, $data)
40.     {
41.         $getLatLong = $this->geocodeAdapter-
    >getLatLong($data);
42.
43.         $inputData = [
44.             'id_user'    => Auth::user()->id,
45.             'kotkab'     => "Surabaya",
46.             'daerah' => $data['daerah'],
47.             'kodepos'    => $data['kodepos'],
48.             'long'       => $getLatLong['lon'],
49.             'lat'        => $getLatLong['lat'],
50.             'blok_nomor'=> $data['blok_nomor'],
51.             'alamat'     => $data['alamat'],
52.             'info_tambahan'=>$data['kotkab']
53.         ];
54.
55.         return $this->model->where('id', $id)-
    >update($inputData);
56.     }
57.
58.     public function tambahAlamat($data)
59.     {
60.         $getLatLong = $this->geocodeAdapter-
    >getLatLong($data);
61.
62.         $inputData = [
63.             'id'         => Uuid::generate()->string,
64.             'id_user'    => Auth::user()->id,
65.             'kotkab'     => "Surabaya",
66.             'daerah' => $data['daerah'],
67.             'kodepos'    => $data['kodepos'],
68.             'long'       => $getLatLong['lon'],
69.             'lat'        => $getLatLong['lat'],
70.             'blok_nomor'=> $data['blok_nomor'],
71.             'alamat'     => $data['alamat'],
```

```
72.             'info_tambahan'=>$data['kotkab']
73.         ];
74.
75.         return $this->model->create($inputData);
76.     }
77. }
78. ?>
```

Kode Sumber 5.20. Repository AlamatRepository

5.2.2. Lapisan Repository BarangKemasanRepository

Lapisan Repository BarangKemasanRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Barangs_kemasan. Kode Sumber 5.21 berikut merupakan implementasi dari lapisan repository BarangKemasanRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.  use App\Repositories\Interfaces\BarangKemasanInterf
    ace;
5.  use App\Models\Barangs_kemasan;
6.
7.  class BarangKemasanRepository implements BarangKema
    sanInterface
8.  {
9.      protected $model;
10.
11.     public function __construct(Barangs_kemasan $mo
    del)
12.     {
13.         $this->model = $model;
14.     }
15.
```

```
16.         public function findByIdBarang($id_barang)
17.         {
18.             return $this->model-
        >where('id_barang', $id_barang)->get();
19.         }
20.     }
21.     ?>
```

Kode Sumber 5.21. Repository BarangKemasanRepository

5.2.3. Lapisan Repository BarangTanggalRepository

Lapisan Repository BarangTanggalRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Barangs_tanggal. Kode Sumber 5.22 berikut merupakan implementasi dari lapisan repository BarangTanggalRepository:

```
1.      <?php
2.
3.      namespace App\Repositories;
4.      use App\Repositories\Interfaces\BarangTanggalInterf
        ace;
5.      use App\User;
6.      use Webpatser\Uuid\Uuid;
7.      use Auth;
8.      use App\Models\Barang_tanggal;
9.
10.     class BarangTanggalRepository implements BarangTang
        galInterface
11.     {
12.         protected $model;
13.
14.         public function __construct(Barang_tanggal $mod
        el)
15.         {
16.             $this->model = $model;
```

```
17.         }
18.
19.         public function create($tanggal, $barang)
20.         {
21.             $data = [
22.                 'id' => Uuid::generate(4),
23.                 'tanggal' => $tanggal,
24.                 'id_barang' => $barang
25.             ];
26.             return $this->model->create($data);
27.         }
28.
29.         public function read()
30.         {
31.             return $this->model->all();
32.         }
33.
34.         public function update($tanggal, $flag)
35.         {
36.             $data = [
37.                 'flag' => $flag
38.             ];
39.             return $this->model->where([
40.                 ['tanggal', $tanggal]
41.                 ])->update($data);
42.         }
43.
44.         public function delete(int $id)
45.         {
46.             return $this->model->find($id)->delete();
47.         }
48.
49.         public function find_tanggal($tanggal)
50.         {
51.             return $this->model->where([
52.                 ['tanggal', $tanggal],
53.                 ['flag',1]
```

```
54.                 ])->orderBy('tanggal')
55.                 ->get();
56.         }
57.
58.         public function find_tanggal_not_status($tangga
    l)
59.         {
60.             return $this->model->where([
61.                 ['tanggal', $tanggal]
62.                 ])->orderBy('tanggal')
63.                 ->get();
64.         }
65.
66.         public function get_tanggal(){
67.             return $this->model->where([
68.                 ['flag', 1]
69.                 ])->orderBy('tanggal')
70.                 ->get();
71.         }
72.
73.         public function deleteByTanggal($tanggal){
74.             return $this->model-
    >where('tanggal',$tanggal)->delete();
75.         }
76.
77.         public function deleteByTanggalKurang($tanggal)
    {
78.             return $this->model-
    >where('tanggal','<=',$tanggal)->delete();
79.         }
80.
81.         public function showEtalaseByTanggal($date){
82.             return $this-
    >model::whereHas('barang', function ($query) use ($
    date) {
83.                 return $query-
    >where('tanggal', '=',$date);
```

```
84.             })->get();
85.         }
86.         public function showByJenisByTanggal($jenis, $d
    ate)
87.         {
88.             return $this-
    >model::whereHas('barang', function ($query) use ($
    date) {
89.                 // dd($jenis);
90.                 return $query-
    >where('tanggal', '=',$date);
91.             })-
    >whereHas('barang', function ($query) use ($jenis)
    {
92.                 // dd($jenis);
93.                 return $query-
    >where('jenis', '=',$jenis);
94.             })->get();
95.         }
96.         public function deleteByIdBarang($id_barang){
97.             return $this->model-
    >where('id_barang',$id_barang)->delete();
98.         }
99.     }
100.    ?>
```

Kode Sumber 5.22. Repository BarangTanggalRepository

5.2.4. Lapisan Repository BaseKategoriRepository

Lapisan Repository BaseKategoriRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Base_kategori. Kode Sumber 5.23 berikut merupakan implementasi dari lapisan repository BaseKategoriRepository:

```
1.  <?php
```

```
2.
3.      namespace App\Repositories;
4.      use App\Repositories\Interfaces\BaseKategoriInterfa
        ce;
5.      use App\Models\Base_Kategori;
6.
7.      class BaseKategoriRepository implements BaseKategor
        iInterface
8.      {
9.          protected $model;
10.
11.         public function __construct(Base_Kategori $mode
        l)
12.         {
13.             $this->model = $model;
14.         }
15.
16.         public function create(array $data)
17.         {
18.             return $this->model->create($data);
19.         }
20.
21.         public function all()
22.         {
23.             return $this->model->all();
24.         }
25.
26.         public function update(int $id,array $data)
27.         {
28.             return $this->model->find($id)-
        >update($data);
29.         }
30.
31.         public function delete(int $id)
32.         {
33.             return $this->model->find($id)->delete();
34.         }
```

```
35.
36.         public function findById(int $id)
37.         {
38.             return $this->model->find($id);
39.         }
40.
41.         public function findByNamaKategori($nama)
42.         {
43.             return $this->model-
    >where('kategori', 'like', '%'.$nama.'%')-
    >first();
44.         }
45.     }
46.     ?>
```

Kode Sumber 5.23. Repository BaseKategoriRepository

5.2.5. Lapisan Repository BobotKemasanRepository

Lapisan Repository BobotKemasanRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Bobot_kemasan. Kode Sumber 5.24 berikut merupakan implementasi dari lapisan repository BobotKemasanRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.  use App\Repositories\Interfaces\BobotKemasanInterfa
    ce;
5.  use App\Models\Bobot_Kemasan;
6.
7.  class BobotKemasanRepository implements BobotKemasa
    nInterface
8.  {
9.      protected $model;
10.
```

```
11.         public function __construct(Bobot_Kemasan $mode
    l)
12.         {
13.             $this->model = $model;
14.         }
15.
16.         public function create(array $data)
17.         {
18.             return $this->model->create($data);
19.         }
20.
21.         public function all()
22.         {
23.             return $this->model->all();
24.         }
25.
26.         public function update(int $id,array $data)
27.         {
28.             return $this->model->find($id)-
    >update($data);
29.         }
30.
31.         public function delete(int $id)
32.         {
33.             return $this->model->find($id)->delete();
34.         }
35.
36.     }
37.     ?>
```

Kode Sumber 5.24. Repository BobotKemasanRepository

5.2.6. Lapisan Repository DetailKeranjangRepository

Lapisan Repository DetailKeranjangRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Detail_keranjang.

Kode Sumber 5.25 berikut merupakan implementasi dari lapisan repository DetailKeranjangRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.  use App\Repositories\Interfaces\DetailKeranjangInte
    rface;
5.  use App\Models\Detail_keranjang;
6.  use Webpatser\Uuid\Uuid;
7.  use Auth;
8.
9.  class DetailKeranjangRepository implements DetailKe
    ranjangInterface
10. {
11.     protected $model;
12.
13.     public function __construct(Detail_keranjang $m
    odel)
14.     {
15.         $this->model = $model;
16.     }
17.
18.     public function findById($id)
19.     {
20.         return $this->model->where('id', $id)-
    >first();
21.     }
22.
23.     public function getDetailKeranjang($id)
24.     {
25.         return $this->model-
    >where('id_keranjang', $id)->get();
26.     }
27.
28.     public function tambahDetailKeranjang($data)
29.     {
```

```
30.             return $this->model->create($data);
31.         }
32.
33.         public function updateDetailKeranjang($id, $dat
    a)
34.         {
35.             return $this->model-
    >where('id_barang',$id)->update($data);
36.         }
37.
38.         public function updateDetailKeranjangKemas($id,
    $bobot, $data)
39.         {
40.             return $this->model
41.                 ->where('id_barang',$id)
42.                 ->where('bobot_kemasan', $bobot)
43.                 ->update($data);
44.         }
45.
46.         public function hapusDetailKeranjang($id)
47.         {
48.             return $this->model->where('id', $id)-
    >delete();
49.         }
50.
51.         public function hapusDetailKeranjangByIdKeranja
    ng($id)
52.         {
53.             return $this->model-
    >where('id_keranjang', $id)->delete();
54.         }
55.
56.         public function cekDetailKeranjang($id)
57.         {
58.             return $this->model-
    >where('id_barang', $id)->first();
59.         }
```

```
60.
61.         public function findByIdBarangBobotKemasan($id_
        keranjang, $id_barang, $bobot)
62.         {
63.             return $this->model-
        >where('id_keranjang', $id_keranjang)-
        >where('id_barang', $id_barang)-
        >where('bobot_kemasan', $bobot)->first();
64.         }
65.
66.         public function findByIdKeranjangIdBarang($id_k
        eranjang,$id_barang)
67.         {
68.             return $this->model-
        >where('id_keranjang', $id_keranjang)-
        >where('id_barang', $id_barang)->first();
69.         }
70.         public function findByIdBarang($id_barang){
71.             return $this->model-
        >where('id_barang',$id_barang)->get();
72.         }
73.         public function deleteByIdBarang($id_barang){
74.             return $this->model-
        >where('id_barang',$id_barang)->delete();
75.         }
76.
77.     }
78.
79.     ?>
```

Kode Sumber 5.25. Repository DetailKeranjangRepository

5.2.7. Lapisan Repository DetailKlaimRepository

Lapisan Repository DetailKlaimRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Detail_klaim.

Kode Sumber 5.26 berikut merupakan implementasi dari lapisan repository DetailKlaimRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.  use App\Repositories\Interfaces\DetailKlaimInterfac
    e;
5.  use App\Models\Detail_klaim;
6.  use Uuid;
7.
8.  class DetailKlaimRepository implements DetailKlaimI
    nterface
9.  {
10.     protected $model;
11.
12.     public function __construct(Detail_klaim $model
    )
13.     {
14.         $this->model = $model;
15.     }
16.
17.     public function create($data)
18.     {
19.         $data['id'] = Uuid::generate()->string;
20.         return $this->model->create($data);
21.     }
22.
23.     public function all()
24.     {
25.         return $this->model->all();
26.     }
27.
28.     public function update(int $id,array $data)
29.     {
30.         return $this->model->find($id)-
    >update($data);
```

```
31.         }
32.
33.         public function delete(int $id)
34.         {
35.             return $this->model->find($id)->delete();
36.         }
37.
38.         public function getDetailKlaimById($id){
39.             return $this->model->where('id_klaim',$id)-
    >get()->all();
40.         }
41.         public function getDetailKlaimByIdDetailTransak
    si($id){
42.             return $this->model-
    >where('id_detail_transaksi',$id)->get();
43.         }
44.         public function findByIdBarang($id_barang){
45.             return $this->model-
    >where('id_barang',$id_barang)->get();
46.         }
47.         public function deleteByIdBarang($id_barang){
48.             return $this->model-
    >where('id_barang',$id_barang)->delete();
49.         }
50.     }
51.     ?>
```

Kode Sumber 5.26. Repository DetailKlaimRepository

5.2.8. Lapisan Repository DetailTransaksiRepository

Lapisan Repository DetailTransaksiRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Detail_transaksi. Kode Sumber 5.27 berikut merupakan implementasi dari lapisan repository DetailTransaksiRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.  use App\Models\Detail_transaksi as DetailTransaksi;
5.  use App\Repositories\Interfaces\DetailTransaksiInterface;
6.  use Carbon\Carbon;
7.  use Webpatser\Uuid\Uuid;
8.  use Illuminate\Support\Facades\Auth;
9.
10. class DetailTransaksiRepository implements DetailTransaksiInterface
11. {
12.     protected $model;
13.
14.     public function __construct(DetailTransaksi $model)
15.     {
16.         $this->model = $model;
17.     }
18.
19.     public function inputDetailTransaksi($data)
20.     {
21.         $dataInput = [
22.             'id'           => Uuid::generate()->string,
23.             'id_barang'    => $data['id_barang'],
24.             'harga'        => $data['harga'],
25.             'harga_diskon'=> $data['harga_diskon'],
26.             'volume'       => $data['volume'],
27.             'id_transaksi'=> $data['id_transaksi'],
28.             'bobot_kemasan'=> $data['bobot_kemasan']
29.         ];
```

```
30.
31.             return $this->model->create($dataInput);
32.         }
33.
34.         public function findByTransaksiId($transaksiId)

35.         {
36.             return $this->model-
        >where('id_transaksi',$transaksiId)->get();
37.         }
38.
39.         public function excludeDetailPreOrder($id)
40.         {
41.                 $data = $this->model->find($id);
42.                 $data->is_exclude_rekap = 1;
43.                 $data->save();
44.         }
45.
46.         public function cancelDetailPreOrder($id)
47.         {
48.                 $data = $this->model->find($id);
49.                 $data->is_canceled_by_veggo = 1;
50.                 $data->save();
51.         }
52.
53.         public function detailOrderKePetani($data){
54.
55.             $harga_beli_terkecil = ($data['barang']-
        >harga_beli / $data['barang']->bobot);
56.             $harga = $harga_beli_terkecil * $data['akum
        ulasi_barang']->volume * $data['akumulasi_barang']-
        >bobot;
57.
58.             $dataInput = [
59.                 'id'           => Uuid::generate()-
        >string,
60.                 'id_barang'    => $data['barang']->id,
```

```
61.                 'harga'        => $harga,
62.                 'status'        => 1,
63.                 'volume'       => $data['akumulasi_baran
    g']->volume,
64.                 'id_transaksi'=> $data['id_transaksi'],

65.                 'bobot_kemasan'=> $data['akumulasi_bara
    ng']->bobot,
66.             ];
67.             return $this->model->create($dataInput);
68.         }
69.
70.         public function updatePengirimanPetani($id,$val
    ue,$value2, $value3, $keterangan,$flag){
71.             $detail = $this->model->find($id);
72.             $detail->volume_kirim_petani = $value;
73.             $detail->bobot_kirim_petani = $value2;
74.             $detail->status = $flag;
75.             $detail->volume_selisih = ($value-$detail-
    >volume);
76.             $detail->bobot_selisih = ($value2-$detail-
    >bobot_kemasan);
77.             $detail->selisih_kirim = $value3;
78.             $detail->keterangan = $keterangan;
79.             return $detail->save();
80.         }
81.
82.         public function updatePenerimaanOrderKePetani($
    id,$value, $value2, $value3,$flag){
83.             $detail = $this->model->find($id);
84.             $detail->volume_terima = $value;
85.             $detail->bobot_terima=$value2;
86.             $detail->selisih_terima=$value3;
87.             $detail->status = $flag;
88.             return $detail->save();
89.         }
90.
```

```
91.         public function updateFinalisasiPengiriman($id,
        $volume, $bobot,$harga, $harga_diskon){
92.             $detail = $this->model->find($id);
93.             $detail->volume_kirim_kurir = $volume;
94.             $detail->bobot_kirim_kurir = $bobot;
95.             $detail->harga_akhir = $harga;
96.             $detail-
        >harga_akhir_diskon = $harga_diskon;
97.             $detail->status = 4;
98.             return $detail->save();
99.         }
100.
101.        public function update($id,$data)
102.        {
103.            return $this->model->find($id)-
        >update($data);
104.        }
105.
106.        public function updateByIdTransaksi($id,$data)

107.        {
108.            return $this->model-
        >where('id_transaksi',$id)->update($data);
109.        }
110.
111.        public function addFinalisasiItem($id_transaksi
        ,$id_barang,$jumlah_kirim,$harga){
112.            $dataInput = [
113.                'id'            => Uuid::generate()-
        >string,
114.                'id_barang'     => $id_barang,
115.                'harga'         => $harga,
116.                'harga_akhir'   => $harga,
117.                'status'        => 1,
118.                'volume'        => $jumlah_kirim,
119.                'id_transaksi'  => $id_transaksi,
120.                'bobot_kemasan' => 1,
```

```
121.                 'volume_kirim_kurir'=> $jumlah_kirim,
122.             ];
123.
124.             $this->model->create($dataInput);
125.         }
126.
127.         public function removeFinalisasiItem($id_detail
    _transaksi){
128.             $detailTransaksi = $this->model-
    >find($id_detail_transaksi);
129.             $detailTransaksi->status = 7;
130.             return $detailTransaksi->save();
131.         }
132.
133.         public function findByIdBarang($id_barang){
134.             return $this->model-
    >where('id_barang',$id_barang)->get();
135.         }
136.         public function deleteByIdBarang($id_barang){
137.             return $this->model-
    >where('id_barang',$id_barang)->delete();
138.         }
139.
140.     }
141.     ?>
```

Kode Sumber 5.27. Repository DetailTransaksiRepository

5.2.9. Lapisan Repository FotoProdukRepository

Lapisan Repository FotoProdukRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Foto_barang. Kode Sumber 5.28 berikut merupakan implementasi dari lapisan repository FotoProdukRepository:

```
1.      <?php
```

```
2.
3.      namespace App\Repositories;
4.      use App\Repositories\Interfaces\FotoProdukInterface
        ;
5.      use App\Models\Foto_barang;
6.      use Uuid;
7.
8.
9.      class FotoProdukRepository implements FotoProdukInt
        erface
10.     {
11.         protected $model;
12.
13.         public function __construct(Foto_barang $model)

14.         {
15.             $this->model = $model;
16.         }
17.
18.         public function create($id, $photo)
19.         {
20.             $data = [
21.                 'id' => Uuid::generate(4),
22.                 'id_barang' => $id,
23.                 'path' => $photo
24.             ];
25.
26.             return $this->model-
        >create($data);
27.         }
28.
29.         public function all()
30.         {
31.             return $this->model->all();
32.         }
33.
34.
```

```
35.         public function delete($id_barang, $path)
36.         {
37.             $data = $this->model
38.                     ->where('id_barang', $id_barang)
39.                     ->where('path', $path)
40.                     ->first();
41.
42.             return $data->delete();
43.
44.         }
45.
46.         public function findByIdBarang($id_barang)
47.         {
48.             return $this->model-
    >where('id_barang',$id_barang)->get();
49.         }
50.
51.         public function deleteByIdBarang($id_barang){
52.             return $this->model-
    >where('id_barang',$id_barang)->delete();
53.         }
54.     }
55.     ?>
```

Kode Sumber 5.28. Repository FotoProdukRepository

5.2.10. Lapisan Repository InventarisRepository

Lapisan Repository InventarisRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Inventaris. Kode Sumber 5.29 berikut merupakan implementasi dari lapisan repository InventarisRepository:

```
1.      <?php
2.
3.      namespace App\Repositories;
```

```
4.  use App\Repositories\Interfaces\InventarisInterface
    ;
5.  use App\Models\Inventaris;
6.  use Uuid;
7.  use Carbon\Carbon;
8.
9.  class InventarisRepository implements InventarisInt
    erface
10. {
11.     protected $model;
12.
13.     public function __construct(Inventaris $model)

14.     {
15.         $this->model = $model;
16.     }
17.
18.     public function create($data)
19.     {
20.         return $this->model->create($data);
21.     }
22.
23.     public function all()
24.     {
25.         return $this->model->all();
26.     }
27.
28.     public function update(int $id,array $data)
29.     {
30.         return $this->model->find($id)-
    >update($data);
31.     }
32.
33.     public function delete(int $id)
34.     {
35.         return $this->model->find($id)->delete();
36.     }
```

```
37.
38.         public function inventarisIn($id_barang,$id_tra
    nsaksi,$value,$keterangan)
39.         {
40.             $data = [
41.                 'id' => Uuid::generate()->string,
42.                 'id_barang' => $id_barang,
43.                 'id_transaksi' => $id_transaksi,
44.                 'tanggal' => Carbon::now(),
45.                 'status' => 'IN',
46.                 'keterangan' => $keterangan,
47.                 'jumlah' => $value
48.             ];
49.
50.             $this->create($data);
51.         }
52.
53.         public function inventarisOut($id_barang,$id_tr
    ansaksi,$value,$keterangan)
54.         {
55.             $data = [
56.                 'id' => Uuid::generate()->string,
57.                 'id_barang' => $id_barang,
58.                 'id_transaksi' => $id_transaksi,
59.                 'tanggal' => Carbon::now(),
60.                 'status' => 'OUT',
61.                 'keterangan' => $keterangan,
62.                 'jumlah' => $value
63.             ];
64.
65.             $this->create($data);
66.         }
67.         public function deleteByIdBarang($id_barang){
68.             return $this->model-
    >where('id_barang',$id_barang)->delete();
69.         }
70.     }
```

```
71.     ?>
```

Kode Sumber 5.29. Repository InventarisRepository

5.2.11. Lapisan Repository IsiPaketRepository

Lapisan Repository IsiPaketRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Isi_paket. Kode Sumber 5.30 berikut merupakan implementasi dari lapisan repository IsiPaketRepository:

```
1.      <?php
2.
3.      namespace App\Repositories;
4.      use App\Repositories\Interfaces\IsiPaketInterface;
5.      use App\Models\Isi_paket;
6.      use Uuid;
7.      use DB;
8.
9.      class IsiPaketRepository implements IsiPaketInterfa
        ce
10.     {
11.         protected $model;
12.
13.         public function __construct(Isi_paket $model)
14.         {
15.             $this->model = $model;
16.         }
17.
18.         public function create($id,$items)
19.         {
20.             if($this->model-
        >where('id_barang_parent',$id) != null){
21.                 $this->model-
        >where('id_barang_parent',$id)->delete();
```

```
            }

            foreach($items['isi_paket'] as $key => $item){

                $data = [
                    'id' => Uuid::generate(4),
                    'id_barang_parent' => $id,
                    'id_barang' => $item,
                    'volume' => $items['volume_isi_paket'][$key],
                    'harga' => $items['harga_isi_paket'][$key],
                ];

                $this->model->create($data);
            }

        }

        public function findByIdBarang($id_barang)
        {
            return $this->model->where('id_barang_parent', $id_barang)->get();
        }

        public function all()
        {
            return $this->model->all();
        }

        public function update($id,$data)
        {
            return $this->model->find($id)->update($data);
        }

```

```
54.         public function delete($id)
55.         {
56.             return $this->model->find($id)->delete();
57.         }
58.
59.         public function read($id_barang_parent)
60.         {
61.             return DB::select('call sp_get_isi_paket(?)
    ',array($id_barang_parent));
62.         }
63.         public function deleteByIdBarang($id_barang){
64.             return $this->model-
    >where('id_barang',$id_barang)->delete();
65.         }
66.         public function deleteByIdParentBarang($id_bara
    ng){
67.             return $this->model-
    >where('id_barang_parent',$id_barang)->delete();
68.         }
69.     }
70.     ?>
```

Kode Sumber 5.30. Repository InventarisRepository

5.2.12. Lapisan Repository KategoriRepository

Lapisan Repository KategoriRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Kategori. Kode Sumber 5.31 berikut merupakan implementasi dari lapisan repository KategoriRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.  use App\Repositories\Interfaces\KategoriInterface;
```

```
5.  use App\Models\Kategori;
6.
7.  class KategoriRepository implements KategoriInterfa
    ce
8.  {
9.      protected $model;
10.
11.     public function __construct(Kategori $model)
12.     {
13.         $this->model = $model;
14.     }
15.
16.     public function create(array $data)
17.     {
18.         return $this->model->create($data);
19.     }
20.
21.     public function all()
22.     {
23.         return $this->model->all();
24.     }
25.
26.     public function findById(int $id)
27.     {
28.         return $this->model->find($id);
29.     }
30.
31.     public function update($id,array $data)
32.     {
33.         return $this->model->find($id)-
    >update($data);
34.     }
35.
36.     public function delete($id)
37.     {
38.         return $this->model->find($id)->delete();
39.     }
```

```
40.
41.         public function array()
42.         {
43.             $kategori = [
44.                 'sayur' => $this->kategoriSayur(),
45.                 'buah' => $this->kategoriBuah(),
46.                 'makanansehat' => $this-
    >kategoriMakananSehat(),
47.                 'minumansehat' => $this-
    >kategoriMinumanSehat(),
48.                 'beras' => $this->kategoriBeras(),
49.                 'bahanolahan' => $this-
    >kategoriBahanOlahan(),
50.                 'berkebun' => $this-
    >kategoriBerkebun(),
51.                 'lainlain' => $this-
    >kategoriLainLain()
52.             ];
53.             return $kategori;
54.         }
55.
56.         public function getNamaKategori(){
57.             return $this->model-
    >select('sub_kategori')->get();
58.         }
59.
60.         public function kategoriSayur()
61.         {
62.             return $this->model->where('kategori', 1)-
    >get();
63.         }
64.
65.         public function kategoriBuah()
66.         {
67.             return $this->model->where('kategori', 2)-
    >get();
68.         }
```

```
69.
70.         public function kategoriMakananSehat()
71.         {
72.             return $this->model->where('kategori', 5)-
    >get();
73.         }
74.
75.         public function kategoriMinumanSehat()
76.         {
77.             return $this->model->where('kategori', 6)-
    >get();
78.         }
79.
80.         public function kategoriBeras()
81.         {
82.             return $this->model->where('kategori', 3)-
    >get();
83.         }
84.
85.         public function kategoriBahanOlahan()
86.         {
87.             return $this->model->where('kategori', 4)-
    >get();
88.         }
89.
90.         public function kategoriBerkebun()
91.         {
92.             return $this->model->where('kategori', 7)-
    >get();
93.         }
94.
95.         public function kategoriLainLain()
96.         {
97.             return $this->model->where('kategori', 8)-
    >get();
98.         }
99.     }
```

```
100.  ?>
```

Kode Sumber 5.31. Repository KategoriRepository

5.2.13. Lapisan Repository KeranjangRepository

Lapisan Repository KeranjangRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Keranjang. Kode Sumber 5.32 berikut merupakan implementasi dari lapisan repository KeranjangRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.  use App\Repositories\Interfaces\KeranjangInterface;
5.  use App\Models\Keranjang;
6.  use Carbon\Carbon;
7.  use Webpatser\Uuid\Uuid;
8.  use Illuminate\Support\Facades\Auth;
9.
10. class KeranjangRepository implements KeranjangInterfa
      ce
11. {
12.     protected $model;
13.
14.     public function __construct(Keranjang $model)
15.     {
16.         $this->model = $model;
17.     }
18.
19.     public function getKeranjang($id)
20.     {
21.         return $this->model-
      >where('id_keranjang', $id)->get();
22.     }
23.
```

```
24.     public function getIDKeranjang($tanggal)
25.     {
26.         // dd($tanggal);
27.         return $this->model->where('id_user', Auth::user()->id)->where('tanggal_pre_order', $tanggal)->first()->id;
28.     }
29.
30.     public function getKeteranganKeranjang($date)
31.     {
32.         return $this->model->where('id_user', Auth::user()->id)->where('tanggal_pre_order', $date)->first();
33.     }
34.
35.     public function tambahKeranjang($tanggal)
36.     {
37.         $inputData = [
38.             'id'                => Uuid::generate(4)->string,
39.             'id_user'           => Auth::user()->id,
40.             'tanggal_pre_order' => $tanggal
41.         ];
42.
43.         return $this->model->create($inputData);
44.     }
45.
46.     public function updateKeranjang($id, $data)
47.     {
48.         #tanggal pre order
49.         $date = Carbon::parse($data['tanggal_pre_order']);
50.
51.         $inputData = [
52.             'tanggal_pre_order' => $date
```

```
53.             ];
54.
55.             return $this->model->where('id',$id)-
    >update($data);
56.         }
57.
58.         public function hapusKeranjang($id)
59.         {
60.             return $this->model->where('id', $id)-
    >delete();
61.         }
62.
63.         public function hapusKeranjangByID($id)
64.         {
65.             return $this->model-
    >where('id_user', Auth::user()->id)->delete();
66.         }
67.
68.         public function findKeranjang($tanggal)
69.         {
70.             return $this->model-
    >where('id_user', Auth::user()->id)
71.             ->where('tanggal_pre_order', $tanggal)-
    >first();
72.         }
73.     }
74.     ?>
```

Kode Sumber 5.32. Repository KeranjangRepository

5.2.14. Lapisan Repository KlaimRepository

Lapisan Repository KlaimRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Klaim. Kode Sumber 5.33 berikut merupakan implementasi dari lapisan repository KlaimRepository:

```php
<?php

namespace App\Repositories;
use App\Repositories\Interfaces\KlaimInterface;
use App\Models\Klaim;
use Uuid;
use Carbon\Carbon;

class KlaimRepository implements KlaimInterface
{
    protected $model;

    public function __construct(Klaim $model)
    {
        $this->model = $model;
    }

    public function create(array $data)
    {

        #tanggal pre order
        $date = Carbon::now();

        #generate nomor invoice
        $rand           = substr(uniqid('', true), -5);
        $kode_klaim  = 'KL' . $rand;

        $data['id'] = Uuid::generate()->string;
        $data['kode_klaim'] = $kode_klaim;

        return $this->model->create($data);
    }

    public function all()
    {
```

```
36.             return $this->model->orderBy('created_at', 'DESC')->get()->all();
37.         }
38.
39.         public function update(int $id,array $data)
40.         {
41.             return $this->model->find($id)->update($data);
42.         }
43.
44.         public function delete(int $id)
45.         {
46.             return $this->model->find($id)->delete();
47.         }
48.
49.         public function getById($id){
50.             return $this->model->where('klaim_to', $id)->get()->all();
51.         }
52.
53.         public function getByIdTrx($id){
54.             return $this->model->where('id', $id)->get()->all();
55.         }
56.     }
57.     ?>
```

Kode Sumber 5.33. Repository KlaimRepository

5.2.15. Lapisan Repository ProdukGroupRepository

Lapisan Repository ProdukGroupRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Barangs_group. Kode Sumber 5.34 berikut merupakan implementasi dari lapisan repository ProdukGroupRepository:

```
1.     <?php
```

```
2.
3.      namespace App\Repositories;
4.      use App\Repositories\Interfaces\ProdukGroupInterfac
        e;
5.      use App\Models\Barangs_group;
6.      use Uuid;
7.
8.      class ProdukGroupRepository implements ProdukGroupI
        nterface
9.      {
10.         protected $model;
11.
12.         public function __construct(Barangs_group $mode
        l)
13.         {
14.             $this->model = $model;
15.         }
16.
17.
18.         public function all()
19.         {
20.             return $this->model->all();
21.         }
22.
23.         public function delete($id)
24.         {
25.             return $this->model-
        >where('id_barang',$id)->delete();
26.         }
27.
28.         public function create($id, $groups)
29.         {
30.             if($this->model->find($id) != null){
31.                 $this->delete($id);
32.             }
33.
34.             foreach($groups as $group){
```

```
35.
36.                 $data = [
37.                     'id' => Uuid::generate(4),
38.                     'id_barang' => $id,
39.                     'id_kategori' => $group
40.                 ];
41.
42.                 $this->model->create($data);
43.             }
44.         }
45.
46.         public function findByIdBarang($id_barang)
47.         {
48.             return $this->model-
    >where('id_barang',$id_barang)->get();
49.         }
50.         public function findByIdKategori($id)
51.         {
52.             return $this->model-
    >where('id_kategori',$id)->get();
53.         }
54.         public function deleteByIdBarang($id_barang){
55.             return $this->model-
    >where('id_barang',$id_barang)->delete();
56.         }
57.     }
58.     ?>
```

Kode Sumber 5.34. Repository ProdukGroupRepository

5.2.16. Lapisan Repository ProdukKemasanRepository

Lapisan Repository ProdukKemasanRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Barangs_kemasan. Kode Sumber 5.35 berikut

merupakan implementasi dari lapisan repository ProdukKemasanRepository:

```
1.  <?php
2.
3.  namespace App\Repositories;
4.  use App\Repositories\Interfaces\ProdukKemasanInterfac
    e;
5.  use App\Models\Barangs_kemasan;
6.  use Uuid;
7.
8.  class ProdukKemasanRepository implements ProdukKemasa
    nInterface
9.  {
10.     protected $model;
11.
12.     public function __construct(Barangs_kemasan $mode
    l)
13.     {
14.         $this->model = $model;
15.     }
16.
17.     public function all()
18.     {
19.         return $this->model->all();
20.     }
21.
22.     public function delete($id){
23.         return $this->model->where('id_barang',$id)-
    >delete();
24.     }
25.
26.     public function create($id, $bobot_kemasan)
27.     {
28.
29.         if($this->model->find($id) != null){
30.             $this->delete($id);
```

```
31.             }
32.
33.             foreach($bobot_kemasan as $bobot){
34.
35.                 $data = [
36.                     'id' => Uuid::generate(4),
37.                     'id_barang' => $id,
38.                     'bobot_kemasan' => $bobot
39.                 ];
40.
41.                 $this->model->create($data);
42.             }
43.         }
44.
45.         public function findByIdBarang($id_barang)
46.         {
47.             return $this->model-
    >where('id_barang',$id_barang)->get();
48.         }
49.
50.         public function deleteByIdBarang($id_barang){
51.             return $this->model-
    >where('id_barang',$id_barang)->delete();
52.         }
53.     }
54.     ?>
```

Kode Sumber 5.35. Repository ProdukKemasanRepository

5.2.17. Lapisan Repository ProdukRepository

Lapisan Repository ProdukRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Barang. Kode Sumber 5.36 berikut merupakan implementasi dari lapisan repository ProdukRepository:

```
<?php

namespace App\Repositories;
use App\Repositories\Interfaces\ProdukInterface;
use App\Models\Barang;
use Uuid;
use DB;


class ProdukRepository implements ProdukInterface
{
    protected $model;

    public function __construct(Barang $model)
    {
        $this->model = $model;
    }

    public function create($data)
    {
        return $this->model->create($data);
    }

    public function create_paket($data)
    {
        return $this->model->create($data);
    }

    public function all()
    {
        return $this->model->all();
    }

    public function getBarangTanpaPenjual(){
        return $this->model->where('id_user', '!=', '3dceb0cc-9983-470e-9e2b-67facc51175d')
```

```
36.                                 ->get()->all();
37.         }
38.
39.         public function showEtalase()
40.         {
41.             return $this->model-
    >where('show_etalase', 1)->get();
42.         }
43.
44.         public function showByJenis($jenis)
45.         {
46.             return $this->model-
    >where('show_etalase', 1)->where('jenis', $jenis)-
    >get();
47.         }
48.
49.         public function update($id,$data)
50.         {
51.             return $this->model->find($id)-
    >update($data);
52.         }
53.
54.         public function delete($id)
55.         {
56.             return $this->model->find($id)->delete();
57.         }
58.
59.         public function findById($id)
60.         {
61.             return $this->model->find($id);
62.         }
63.
64.         public function findByUserId($id)
65.         {
66.             return $this->model->where('id_user',$id)-
    >get();
67.         }
```

```
68.
69.         public function findByKode($kode)
70.         {
71.             return $this->model->where('kode',$kode)-
    >first();
72.         }
73.
74.         public function findByIsPaket($flag)
75.         {
76.             return $this->model-
    >where('is_paket',$flag)->get();
77.         }
78.
79.         public function findByIdKategori($id_kategori)

80.         {
81.             return $this->model->where([
82.                 ['id_kategori',$id_kategori],
83.                 ['show_etalase', 1]
84.                 ])->get();
85.         }
86.
87.         public function findByIdKategoriOrNama($id_kate
    gori,$nama)
88.         {
89.             return $this->model-
    >where('id_kategori',$id_kategori)-
    >where('nama','like','%'.$nama.'%')->get();
90.         }
91.
92.         public function findByNamaSubKategori($nama, $t
    anggal)
93.         {
94.             return DB::select("CALL sp_get_barang_by_ka
    tegori(?, ?)", array($nama, $tanggal));
95.         }
96.
```

```
97.         public function findByNamaBarang($nama)
98.         {
99.             return $this->model-
      >where('nama', 'like', '%'.$nama.'%')->get();
100.        }
101.
102.        public function addStok($id,$value){
103.            $produk = $this->model->find($id);
104.            $produk->stok += $value;
105.            return $produk->save();
106.        }
107.
108.        public function removeStok($id,$value){
109.            $produk = $this->model->find($id);
110.            $produk->stok -= $value;
111.            return $produk->save();
112.        }
113.
114.        public function getProduct(){
115.            return $this->model-
      >where('jenis', '!=', 'Paket')->get()->all();
116.        }
117.        public function getPaket(){
118.            return $this->model-
      >where('jenis', '=', 'Paket')->get()->all();
119.        }
120.    }
121.    ?>
```

Kode Sumber 5.36. Repository ProdukRepository

5.2.18. Lapisan Repository TanggalRepository

Lapisan Repository TanggalRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Tanggal. Kode Sumber 5.37 berikut merupakan implementasi dari lapisan repository TanggalRepository:

```php
<?php

namespace App\Repositories;
use App\Repositories\Interfaces\TanggalInterface;
use App\User;
use Webpatser\Uuid\Uuid;
use Auth;
use App\Models\Tanggal;

class TanggalRepository implements TanggalInterface
{
    protected $model;

    public function __construct(Tanggal $model)
    {
        $this->model = $model;
    }

    public function create($tanggal, $flag)
    {
        $data = [
            'id' => Uuid::generate(4),
            'tanggal' => $tanggal,
            'flag' => $flag
        ];
        return $this->model->create($data);
    }

    public function read()
    {
        return $this->model->all();
    }

    public function update($tanggal, $flag)
    {
        $data = [
```

```
37.                 'flag' => $flag
38.             ];
39.             return $this->model->where([
40.                 ['tanggal', $tanggal]
41.                 ])->update($data);
42.         }
43.
44.         public function delete(int $id)
45.         {
46.             return $this->model->find($id)->delete();
47.         }
48.
49.         public function hapusTanggal($tanggal)
50.         {
51.             $data = [
52.                 'flag' => 0
53.             ];
54.             return $this->model->where([
55.                 ['tanggal', '<=', $tanggal]
56.                 ])->update($data);
57.         }
58.         public function find_tanggal($tanggal)
59.         {
60.             return $this->model->where([
61.                 ['tanggal', $tanggal],
62.                 ['flag',1]
63.                 ])->orderBy('tanggal')
64.                 ->get();
65.         }
66.
67.         public function find_tanggal_not_status($tangga
    l)
68.         {
69.             return $this->model->where([
70.                 ['tanggal', $tanggal]
71.                 ])->orderBy('tanggal')
72.                 ->get();
```

```
73.         }
74.
75.         public function get_tanggal(){
76.             return $this->model->where([
77.                 ['flag', 1]
78.                 ])->orderBy('tanggal')
79.                 ->get();
80.         }
81.
82.
83.     }
84.     ?>
```

Kode Sumber 5.37. Repository TanggalRepository

5.2.19. Lapisan Repository TransaksiRepository

Lapisan Repository TransaksiRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* Transaksi. Kode Sumber 5.38 berikut merupakan implementasi dari lapisan repository TransaksiRepository:

```
1.      <?php
2.
3.      namespace App\Repositories;
4.
5.      use App\Repositories\Interfaces\TransaksiInterface;

6.      use App\Models\Transaksi;
7.      use Illuminate\Support\Facades\Auth;
8.      use Webpatser\Uuid\Uuid;
9.      use Carbon\Carbon;
10.     use DB;
11.
12.     class TransaksiRepository implements TransaksiInter
        face
```

```
13.    {
14.        protected $model;
15.
16.        public function __construct(Transaksi $model)
17.        {
18.            $this->model = $model;
19.        }
20.
21.        public function getAllTransaksiByUser($id)
22.        {
23.            return $this->model->where('id_user', $id)-
       >get();
24.        }
25.
26.        public function getAllTransaksiByUserAndDate($i
       d, $tanggal)
27.        {
28.            return $this->model->where([
29.                ['id_user', $id],
30.                ['tanggal_pre_order',$tanggal]
31.                ])->orderBy('created_at', 'DESC')-
       >get();
32.        }
33.
34.        public function getAllTransaksiByUserAndNotPay(
       $id)
35.        {
36.            return $this->model->where([
37.                ['id_user', $id],
38.                ['isAlreadyPay','!=', 3],
39.                ['status','<', 7],
40.                ['is_canceled_by_veggo', 0],
41.                ])->orderBy('created_at', 'DESC')-
       >get();
42.        }
43.
```

```
44.         public function getAllTransaksiByUserAndInProce
    ss($id)
45.         {
46.             return $this->model->where([
47.                 ['id_user', $id],
48.                 ['isAlreadyPay', 3],
49.                 ['status', '<', 7],
50.                 ['is_canceled_by_veggo', 0],
51.                 ])->orderBy('created_at', 'DESC')-
    >get();
52.         }
53.
54.         public function getAllTransaksiByUserAndIsFinis
    h($id)
55.         {
56.             return $this->model->where([
57.                 ['id_user', $id],
58.                 ['status', 7],
59.                 ['is_canceled_by_veggo', 0],
60.                 ])->orderBy('created_at', 'DESC')-
    >get();
61.         }
62.
63.         public function getAllTransaksiByUserAndIsCance
    lled($id)
64.         {
65.             return $this->model->where([
66.                 ['id_user', $id],
67.                 ['is_canceled_by_veggo', 1],
68.                 ])->orderBy('created_at', 'DESC')-
    >get();
69.         }
70.
71.         public function updateTransaksi($id_transaksi,
    $status)
72.         {
```

```
73.             return $this->model->where('id', $id_transaksi)->update(['status' => $status]);
74.         }
75.
76.         public function konfirmasiTransaksi($id_transaksi){
77.             return $this->model->where([
78.                 ['id_user', Auth::user()->id],
79.                 ['id', $id_transaksi],
80.                 ])->update(['is_confirm_finish_byuser'=> 1]);
81.         }
82.
83.         public function konfirmasiDiterima($id_transaksi, $nama_penerima, $foto_penerima, $keterangan_penerima, $tanggal_terima)
84.         {
85.             return $this->model->where('id', $id_transaksi)->update([
86.                 'nama_penerima' => $nama_penerima,
87.                 'foto_penerima' => $foto_penerima,
88.                 'keterangan_penerima'    => $keterangan_penerima,
89.                 'tanggal_terima' => $tanggal_terima,
90.                 'status'        => 7,
91.                 'isAlreadyPay' =>3
92.             ]);
93.         }
94.
95.         public function kirimBukti($id_transaksi, $bukti_transfer)
96.         {
97.             return $this->model->where('id', $id_transaksi)->update([
98.                 'bukti_transfer' => $bukti_transfer,
99.                 'isAlreadyPay'    => 1,
```

```
100.            ]);
101.        }
102.
103.        public function all(){
104.            return $this->model->all()-
    >sortByDesc("created_at");
105.        }
106.
107.        public function update($id,$data)
108.        {
109.            return $this->model->find($id)-
    >update($data);
110.        }
111.
112.        public function getTransactionDateWithStatus($s
    tatus, $tipe, $cancel)
113.        {
114.            // $current = Carbon::now()-
    >toDateString();
115.            $from = Carbon::now()->subDays(14)-
    >toDateString();
116.            $to = Carbon::now()->addDays(14)-
    >toDateString();
117.            if($cancel==0){
118.                return $dates = $this->model-
    >select("tanggal_pre_order")
119.                                                -
    >where('status', $status)
120.                                                -
    >where('tipe_transaksi',$tipe)
121.                                                -
    >where('is_canceled_by_veggo',$cancel)
122.                                                -
    >whereBetween("tanggal_pre_order",[$from,$to])-
    >groupby("tanggal_pre_order")->get();
123.            }
124.            else{
```

```
125.             return $dates = $this->model-
    >select("tanggal_pre_order")
126.                                         -
    >where('tipe_transaksi',$tipe)
127.                                         -
    >where('is_canceled_by_veggo',$cancel)
128.                                         -
    >whereBetween("tanggal_pre_order",[$from,$to])-
    >groupby("tanggal_pre_order")->get();
129.         }
130.
131.     }
132.     public function getTransactionDateWithStatus2($
    status,$status2, $tipe)
133.     {
134.         // $current = Carbon::now()-
    >toDateString();
135.         $from = Carbon::now()->subDays(14)-
    >toDateString();
136.         $to = Carbon::now()->addDays(14)-
    >toDateString();
137.         return $dates = $this->model-
    >select("tanggal_pre_order")
138.                                         -
    >whereBetween("status",[$status,$status2])
139.                                         -
    >where('tipe_transaksi',$tipe)
140.                                         -
    >where('is_canceled_by_veggo',0)
141.                                         -
    >whereBetween("tanggal_pre_order",[$from,$to])-
    >groupby("tanggal_pre_order")->get();
142.     }
143.     public function getTransactionDate()
144.     {
145.         // $current = Carbon::now()-
    >toDateString();
```

```
146.          $from = Carbon::now()->subDays(14)-
      >toDateString();
147.          $to = Carbon::now()->addDays(14)-
      >toDateString();
148.          return $dates = $this->model-
      >select("tanggal_pre_order")-
      >whereBetween("tanggal_pre_order",[$from,$to])-
      >groupby("tanggal_pre_order")->get();
149.      }
150.
151.      public function findByTanggalPreOrder($date){
152.          return $this->model-
      >where('tanggal_pre_order',$date)
153.                                  ->where('status',1)
154.                                  ->get();
155.      }
156.
157.      public function findByTanggalPreOrderAndStatusA
      ndTipeTransakisAkumulasi($date,$status,$tipe){
158.          return $this->model-
      >where('tanggal_pre_order',$date)
159.                                  -
      >where('status',$status)
160.                                  -
      >where('tipe_transaksi',$tipe)
161.                                  -
      >where('is_canceled_by_veggo',0)
162.                                  -
      >where('is_exclude_rekap',0)
163.                                  -
      >orderBy('created_at', 'DESC')
164.                                  ->get();
165.      }
166.
167.      public function findByTanggalPreOrderAndStatusA
      ndTipeTransakis($date,$status,$tipe){
```

```
168.            return $this->model-
    >where('tanggal_pre_order',$date)
169.                                -
    >where('status',$status)
170.                                -
    >where('tipe_transaksi',$tipe)
171.                                -
    >where('is_canceled_by_veggo',0)
172.                                -
    >orderBy('created_at', 'DESC')
173.                                ->get();
174.        }
175.        public function findByTanggalPreOrderAndStatusA
    ndTipeTransakis2($date,$status,$status2,$tipe){
176.            return $this->model-
    >where('tanggal_pre_order',$date)
177.                                -
    >whereBetween('status',[$status, $status2])
178.                                -
    >where('tipe_transaksi',$tipe)
179.                                -
    >where('is_canceled_by_veggo',0)
180.                                -
    >orderBy('created_at', 'DESC')
181.                                ->get();
182.        }
183.
184.        public function getAllTransaksiIsCancelled($dat
    e,$tipe)
185.        {
186.            return $this->model-
    >where('tanggal_pre_order',$date)
187.                                -
    >where('tipe_transaksi',$tipe)
188.                                -
    >where('is_canceled_by_veggo',1)
```

```
                                ->orderBy('created_at', 'DESC')
                                ->get();
        }

        public function findByTanggalPreOrderAndTipeTransaksi($date,$tipe){
            return $this->model->where('tanggal_pre_order',$date)
                                ->where('status',1)
                                ->where('tipe_transaksi',$tipe)
                                ->get();
        }

        public function excludeRekapPreOrder($id)
        {
            $data = $this->model->find($id);
            $data->is_exclude_rekap = 1;
            $data->save();
        }

        public function cancelPreOrder($id)
        {
            $data = $this->model->find($id);
            $data->is_canceled_by_veggo = 1;
            $data->save();
        }

        public function halamanCheckout($data)
        {
            #tanggal pre order
            $date = Carbon::parse($data['tanggal_pre_order']);

            #generate nomor invoice
```

```
220.            $rand           = substr(uniqid('', true),
    -5);
221.            $nomor_invoice  = 'VG' . $rand;
222.
223.            $inputData = [
224.                'id'                => Uuid::generate()
    ->string,
225.                'id_user'           => Auth::user()-
    >id,
226.                'nomor_invoice'     => $nomor_invoice,
227.                'total_bayar'       => $data['total_bay
    ar'],
228.                'tanggal_pre_order' => $date,
229.                'isCheckout'        => $data['isCheckou
    t']
230.            ];
231.
232.            return $this->model->create($inputData);
233.        }
234.
235.        public function halamanCheckoutReseller($data)
236.        {
237.            #tanggal pre order
238.            $date = Carbon::parse($data['tanggal_pre_or
    der']);
239.
240.            #generate nomor invoice
241.            $rand           = substr(uniqid('', true),
    -5);
242.            $nomor_invoice  = 'VGRES' . $rand;
243.
244.            $inputData = [
245.                'id'                => Uuid::generate()
    ->string,
```

```
246.                 'id_user'           => Auth::user()->id,
247.                 'nomor_invoice'     => $nomor_invoice,
248.                 'total_bayar'       => $data['total_bayar'],
249.                 'tanggal_pre_order' => $date,
250.                 'isCheckout'        => $data['isCheckout']
251.             ];
252.
253.             return $this->model->create($inputData);
254.         }
255.
256.         public function purchaseCheckout($data, $date)
257.         {
258.
259.             $getTransaksi = $this->checkTransaksi($date);
260.             $dataInput = [
261.                 'isCheckout' => 1,
262.                 'status'     => 1,
263.                 'keterangan' => $data['keterangan'],
264.                 'id_alamat'  => $data['id_alamat']
265.             ];
266.
267.             return $this->model
268.                 ->where('id', $getTransaksi->id)
269.                 ->where('id_user', Auth::user()->id)
270.                 ->update($dataInput);
271.         }
272.
273.         public function getIncludedRekap($date,$flag){
274.             return $this->model->where('is_exclude_rekap',$flag)
```

```
                                -
    >where('tanggal_pre_order',$date)
                                ->get();
        }

        public function checkTransaksi($date)
        {
            return $this->model
                ->where('id_user', Auth::user()->id)
                ->where('isCheckout', 0)
                ->where('tanggal_pre_order', $date)
                ->first();
        }

        public function updateHalamanCheckout($data)
        {
            #tanggal pre order
            $date = Carbon::parse($data['tanggal_pre_or
    der']);

            $inputData = [
                'total_bayar'       => $data['total_bay
    ar'],
                'tanggal_pre_order' => $date,
            ];

            return $this->model
                ->where('id_user', Auth::user()->id)
                ->where('isCheckout', 0)
                ->update($inputData);
        }

        public function getPaketYangAkanDikirim()
        {
            $id_kurir = Auth::user()->id;

            return $this->model
```

```
309.                 ->where('id_kurir', $id_kurir)
310.                 ->where('status', 5)
311.                 ->get();
312.         }
313.
314.         public function getPaketDalamPengiriman()
315.         {
316.             $id_kurir = Auth::user()->id;
317.
318.             return $this->model
319.                 ->where('id_kurir', $id_kurir)
320.                 ->where('status', 6)
321.                 ->get();
322.         }
323.
324.         public function getPaketSelesaiDikirim()
325.         {
326.             $id_kurir = Auth::user()->id;
327.
328.             return $this->model
329.                 ->where('id_kurir', $id_kurir)
330.                 ->where('status', 7)
331.                 ->get();
332.         }
333.         public function orderKePetani($data){
334.             #tanggal pre order
335.             $date = Carbon::parse($data['tanggal']);
336.
337.             #generate nomor invoice
338.             $rand           = substr(uniqid('', true),
      -5);
339.             $nomor_invoice  = 'VGPETANI' . $rand;

340.
341.             $inputData = [
342.                 'id'=>Uuid::generate()->string,
343.                 'id_user'=> $data['id_user'],
```

```
344.                'id_alamat'=> "alamat-petani",
345.                'id_kurir'=> "veggo",
346.                'nomor_invoice'=> $nomor_invoice,
347.                'total_bayar'=> 0,
348.                'status'=> 1,
349.                'bukti_transfer'=> 0,
350.                'tanggal_pre_order'=> $data['tanggal'],
351.                'keterangan'=> "order ke petani",
352.                'tipe_transaksi'=> "FROM_VEGGO",
353.            ];
354.
355.            return $this->model->create($inputData);
356.        }
357.
358.        public function updateStatusOrderKePetani($id,$flag){
359.            $transaksi = $this->model->find($id);
360.            $transaksi->status = $flag;
361.            return $transaksi->save();
362.        }
363.
364.        public function updateTanggalPengiriman($id,$date){
365.            $transaksi = $this->model->find($id);
366.            $transaksi->tanggal_pengiriman = $date;
367.            return $transaksi->save();
368.        }
369.
370.        public function updateTanggalTerima($id,$date){
371.            $transaksi = $this->model->find($id);
372.            $transaksi->tanggal_terima = $date;
373.            return $transaksi->save();
374.        }
375.
376.
```

```
377.
378.       public function updateKurirPengiriman($id,$kuri
       r, $reseller,$flag){
379.           $transaksi = $this->model->find($id);
380.           $transaksi->id_kurir = $kurir;
381.           if($reseller==null){
382.               $transaksi->is_diterima_reseller=0;
383.           }
384.           else{
385.               $transaksi->is_diterima_reseller=1;
386.           }
387.           $transaksi->id_reseller = $reseller;
388.           $transaksi->status = $flag;
389.           $transaksi-
       >tanggal_pengiriman = Carbon::now();
390.           return $transaksi->save();
391.       }
392.
393.       public function findByIdUser($id){
394.           return $this->model->where('id_user',$id)-
       >orderBy('created_at', 'DESC')->get();
395.       }
396.
397.       public function find($id){
398.           return $this->model->find($id);
399.       }
400.
401.       public function findByTipeTransaksi($tipe){
402.           return $this->model-
       >where('tipe_transaksi',$tipe)-
       >orderBy('created_at', 'DESC')->get();
403.
404.       }
405.
406.       public function findByNomorInvoice($invoice){
407.           return $this->model-
       >where('nomor_invoice',$invoice)->first();
```

```
408.       }
409.
410.       public function findByTipeTransaksiAndTanggalPr
       eOrderAndStatusOrIsAlreadyPay($tipe,$tanggal,$statu
       s,$ispaid){
411.           return $this->model-
       >where('is_preorder',$tipe)
412.                                   -
       >where('tanggal_pre_order',$tanggal)
413.                                   -
       >where(function($query) use ($status,$ispaid){
414.                                       $query-
       >orwhere('status',$status)-
       >orwhere('isAlreadyPay',$ispaid);
415.                                   })
416.                                   ->get();
417.       }
418.
419.       public function getTransaksiPengiriman($date){
420.           return $this->model-
       >where('tanggal_pre_order',$date)
421.                                    -
       >where('tipe_transaksi',"FROM_BUYER")
422.                                    ->where(function($q){
423.                                        $q-
       >orwhere('status',4)
424.                                        -
       >orwhere('status',5)
425.                                        -
       >orwhere('status',6);
426.                                   })
427.                                   ->get();
428.       }
429.
```

```
430.      public function getPreOrderAndTanggal($tipe,$da
      te){
431.          return $this->model-
      >where('tanggal_pre_order',$date)
432.                                  -
      >where('tipe_transaksi',"FROM_BUYER")
433.                                  -
      >where('is_preorder',1)
434.                                  -
      >where('is_order_to_petani',0)->get();
435.
436.      }
437.
438.      public function getProsesOrderAndTanggal($tipe,
      $date){
439.          return $this->model-
      >where('tanggal_pre_order',$date)
440.                                  -
      >where('tipe_transaksi',"FROM_BUYER")
441.                                  -
      >where('is_order_to_petani',1)->get();
442.
443.      }
444.
445.      public function getCountVerif($status,$tipe, $c
      ancel){
446.          if($cancel==0){
447.              return $this->model-
      >select("tanggal_pre_order" , DB::raw('count(*) as
      total'))
448.                                  -
      >where('tipe_transaksi',"FROM_BUYER")
449.                                  -
      >where('status',$status)
450.                                  -
      >where('tipe_transaksi',$tipe)
```

```
451.                                 -
    >where('isAlreadyPay',1)
452.                                 -
    >where('is_canceled_by_veggo',$cancel)
453.                                 -
    >groupBy('tanggal_pre_order')
454.                                 ->get();
455.             }
456.             else{
457.                 return $this->model-
    >select("tanggal_pre_order" , DB::raw('count(*) as
    total'))
458.                                 -
    >where('tipe_transaksi',"FROM_BUYER")
459.                                 -
    >where('tipe_transaksi',$tipe)
460.                                 -
    >where('isAlreadyPay',1)
461.                                 -
    >where('is_canceled_by_veggo',$cancel)
462.                                 -
    >groupBy('tanggal_pre_order')
463.                                 ->get();
464.             }
465.
466.         }
467.         public function getCountVerif2($status,$status2
    ,$tipe){
468.                 return $this->model-
    >select("tanggal_pre_order" , DB::raw('count(*) as
    total'))
469.                                 -
    >where('tipe_transaksi',"FROM_BUYER")
470.                                 -
    >whereBetween('status',[$status, $status2])
471.                                 -
    >where('tipe_transaksi',$tipe)
```

```
                                -
    >where('isAlreadyPay',1)
                                -
    >where('is_canceled_by_veggo',0)
                                -
    >groupBy('tanggal_pre_order')
                                ->get();
        }

        public function getTransaksiSudahKirim(){
            return $this->model-
    >where('nomor_invoice', 'like', '%VGPETANI%')
                                -
    >whereNotNull('tanggal_pengiriman')
                                ->get()->all();
        }

        public function getTglPetani(){
            $from = Carbon::now()->subDays(14)-
    >toDateString();
            $to = Carbon::now()->addDays(14)-
    >toDateString();

            return $dates = $this->model-
    >select("tanggal_pre_order")
                                        -
    >where('nomor_invoice', 'like', '%VGPETANI%')
                                        -
    >whereBetween("tanggal_pre_order",[$from,$to])-
    >groupby("tanggal_pre_order")->get();
        }

        public function hargaBarangByPetani($date){
            return collect(DB::select('call sp_get_tota
    l_bayar_veggo_by_tanggal(?)',array($date)))-
    >where('id_petani', Auth::user()->id);
        }
```

```
496.      public function getTglByIdPetani(){
497.          $from = Carbon::now()->subDays(14)-
      >toDateString();
498.          $to = Carbon::now()->addDays(14)-
      >toDateString();
499.
500.          return $dates = $this->model-
      >select("tanggal_pre_order")
501.                                          -
      >where('id_user', Auth::user()->id)
502.                                          -
      >whereBetween("tanggal_pre_order",[$from,$to])-
      >groupby("tanggal_pre_order")->get();
503.      }
504.
505.      public function konfirmSampai($confirmReseller,
      $id){
506.          return $this->model->where('id', $id)-
      >update(['is_diterima_reseller' => $confirmReseller
      ]);
507.      }
508.      public function konfirmDiterima($confirmDiterim
      a, $getHariIni, $id){
509.          return $this->model->where('id', $id)
510.                       ->update([
511.                           'status' => $confirmDiterim
      a,
512.                           'tanggal_terima'=>$getHariI
      ni,
513.                           'isAlreadyPay' =>3
514.                           ]);
515.      }
516.
517.      public function getPemasukan($date){
518.          return $this->model-
      >select(DB::raw("ifnull(SUM(total_bayar_akhir), 0)
      as pemasukan"))
```

```
519.            ->where('status', 7)
520.            ->where('tanggal_pre_order', $date)
521.            ->get();
522.        }
523.
524.        public function getTgl(){
525.            $from = Carbon::now()->subDays(14)-
    >toDateString();
526.            $to = Carbon::now()->addDays(14)-
    >toDateString();
527.
528.            return $dates = $this->model-
    >select("tanggal_pre_order")
529.                                        -
    >whereBetween("tanggal_pre_order",[$from,$to])-
    >groupby("tanggal_pre_order")->get();
530.        }
531.        public function getBulanTahun(){
532.            $from = Carbon::now()->subMonths(6)-
    >toDateString();
533.            $to = Carbon::now()->addMonths(6)-
    >toDateString();
534.
535.            return $dates = $this->model-
    >select(DB::raw('DISTINCT EXTRACT(YEAR FROM tanggal
    _pre_order) as tahun,EXTRACT(MONTH FROM tanggal_pre
    _order) as bulan'))
536.                                        -
    >whereBetween("tanggal_pre_order",[$from,$to])
537.                                        -
    >groupby("tanggal_pre_order")
538.                                        ->get();
539.        }
540.        public function getPemasukanTotalHarian($date){
```

```
541.          return $this->model->select(DB::raw('SUM(COALESCE(total_bayar_akhir, 0) ) AS pemasukan'))
542.                                ->where("status",7)
543.                                ->where("tanggal_pre_order", $date)
544.                                ->first();
545.      }
546.      public function getPemasukanTotalBulanan($month, $year){
547.          return $this->model->select(DB::raw('SUM(COALESCE(total_bayar_akhir, 0) ) AS pemasukan'))
548.                                ->whereRaw("status=7 AND EXTRACT(MONTH FROM tanggal_pre_order)=? AND EXTRACT(YEAR FROM tanggal_pre_order)=?", [$month, $year])
549.                                ->first();
550.      }
551.
552.      public function getResellerBydate($date){
553.          $messages = collect($this->model->distinct()
554.                                              ->where("tanggal_pre_order",$date)
555.                                              ->where('nomor_invoice','like', '%VGRES%')
556.                                              ->where('status', 7)
557.                                              ->get()
558.                      );
559.
560.          $messagesUnique = $messages->unique('id_user');
561.          return $messagesUnique->values()->all();
562.      }
563.  }
```

```
564.  ?>
```

Kode Sumber 5.38. Repository TransaksiRepository

5.2.20. Lapisan Repository UserRepository

Lapisan Repository UserRepository merupakan implementasi lapisan *Repository* yang digunakan untuk mengakses *Model* User. Kode Sumber 5.39 berikut merupakan implementasi dari lapisan repository UserRepository:

```
1.    <?php
2.
3.    namespace App\Repositories;
4.    use App\Repositories\Interfaces\UserInterface;
5.    use App\User;
6.    use Auth;
7.
8.    class UserRepository implements UserInterface
9.    {
10.       protected $model;
11.
12.       public function __construct(User $model)
13.       {
14.           $this->model = $model;
15.       }
16.
17.       public function create(array $data)
18.       {
19.           return $this->model->create($data);
20.       }
21.
22.       public function read()
23.       {
24.           return $this->model->all();
25.       }
```

```
26.
27.         public function update(array $data)
28.         {
29.             return $this->model->find(Auth::user()-
    >id)->update($data);
30.         }
31.
32.         public function delete(int $id)
33.         {
34.             return $this->model->find($id)->delete();
35.         }
36.
37.         public function getPetani()
38.         {
39.             return $this->model->where('role',3)-
    >get();
40.         }
41.
42.         public function detail($id)
43.         {
44.             return $this->model->where('id', $id)-
    >first();
45.         }
46.         public function getKurir()
47.         {
48.             return $this->model->where('role',5)-
    >get();
49.         }
50.         public function getVeggo()
51.         {
52.             return $this->model-
    >select('nomor_rek', 'bank', 'atas_nama')-
    >where('role',1)->first();
53.         }
54.         public function getReseller()
55.         {
```

```
56.             return $this->model->where('role',4)-
    >get();
57.         }
58.
59.         public function find($id){
60.             return $this->model->find($id);
61.         }
62.
63.     }
64.     ?>
```

Kode Sumber 5.39. Repository UserRepository

5.3. Implementasi Lapisan Kontrol

Implementasi lapisan kontrol ini berisi logika yang digunakan aplikasi seperti kontrol untuk memodifikasi data, menampilkan data, dan logika lainnya. Di dalam pengembangannya, kami memilih untuk memisahkan mengkategorikan *Controller* berdasarkan role masing masing agar memudahkan *maintenance* dengan mempertahankan *readability*.

5.3.1. Role Pembeli

merupakan kumpulan *Controller* yang dikhususkan untuk role Pembeli. Berikut adalah beberapa *Controller*:

5.3.1.1. AlamatController

Lapisan ini bertugas untuk melakukan sesuau yang berhubungan dengan alamat Pembeli, yaitu menambah alamat, mengubah alamat, dan menghapus alamat. Kode Sumber 5.40 berikut merupakan implementasi dari lapisan kontrol AlamatController:

```php
<?php

namespace App\Http\Controllers\Pembeli;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Repositories\AlamatRepository;
use Illuminate\Support\Facades\DB;

use App\Repositories\KategoriRepository;
use App\Repositories\ProdukRepository;
use App\Repositories\BaseKategoriRepository;
use Illuminate\Support\Facades\Auth;
use URL;

class AlamatController extends Controller
{
    private $alamat;

    public function __construct(AlamatRepository $alamat, KategoriRepository $kategoriRepository, ProdukRepository $produkRepository, BaseKategoriRepository $baseKategoriRepository)
    {
        $this->kategoriRepository = $kategoriRepository;
        $this->produkRepository = $produkRepository;
        $this->baseKategoriRepository = $baseKategoriRepository;

        $this->middleware('auth');
        $this->middleware('pembeli');

        $this->alamat = $alamat;
    }

```

```
31.     public function showAlamat()
32.     {
33.         $data = [
34.             'alamat' => $this->alamat->getAllAlamatByUser(Auth::user()->id)
35.         ];
36.
37.         return view('pembeli.alamat')->with(compact('data'));
38.     }
39.
40.     public function showAlamatbyUser($id)
41.     {
42.         $alamat = [
43.             'alamat' => $this->alamat->getAllAlamatByUser($id)
44.         ];
45.
46.         $kategori = [
47.             'sayur' => $this->kategoriRepository->kategoriSayur(),
48.             'buah' => $this->kategoriRepository->kategoriBuah(),
49.             'makanansehat' => $this->kategoriRepository->kategoriMakananSehat(),
50.             'minumansehat' => $this->kategoriRepository->kategoriMinumanSehat(),
51.             'beras' => $this->kategoriRepository->kategoriBeras(),
52.             'bahanolahan' => $this->kategoriRepository->kategoriBahanOlahan(),
53.             'berkebun' => $this->kategoriRepository->kategoriBerkebun(),
54.             'lainlain' => $this->kategoriRepository->kategoriLainLain()
55.         ];
56.
```

```
57.           $data = [
58.               'barang' => $this->produkRepository-
      >showEtalase(),
59.               'barang_paket' => $this-
      >produkRepository->showByJenis('Paket'),
60.               'barang_timbang' => $this-
      >produkRepository->showByJenis('Timbang'),
61.               'barang_kemas' => $this-
      >produkRepository->showByJenis('Kemas'),
62.               'kategori' => $this->kategoriRepository-
      >array(),
63.               'baseKategori' => $this-
      >baseKategoriRepository->all(),
64.           ];
65.
66.           //dd($alamat);
67.
68.           return view('pembeli.alamat')-
      >with(compact('alamat'))
69.               ->with(compact('data'))
70.               ->with(compact('kategori'));
71.       }
72.
73.       public function hapusAlamat($id)
74.       {
75.           $this->alamat->hapusAlamat($id);
76.
77.           return redirect('Pembeli/LihatAlamat/'.Auth:
      :user()->id);
78.       }
79.
80.       public function _tambahAlamat(Request $request)

81.       {
82.           //dd($request);
83.           try
84.           {
```

```
85.                 DB::beginTransaction();
86.
87.                 $this->alamat->tambahAlamat($request-
    >all());
88.
89.                 DB::commit();
90.
91.                 return redirect(session('url.intended'))
    ;
92.             }
93.             catch (\Throwable $th)
94.             {
95.                 DB::rollback();
96.
97.                 return $th;
98.             }
99.         }
100.
101.        public function tambahAlamat()
102.        {
103.            session()-
    >put('url.intended', URL::previous());
104.            return view('pembeli.tambah-alamat');
105.        }
106.
107.        public function ubahAlamat($id)
108.        {
109.            $alamat=$this->alamat->getAlamatById($id);
110.            return view('pembeli.ubah-alamat')-
    >with(compact('alamat'));
111.        }
112.
113.        public function _ubahAlamat($id, Request $reques
    t)
114.        {
115.            try
116.            {
```

```
117.                DB::beginTransaction();
118.
119.                $this->alamat->ubahAlamat($id, $request-
    >all());
120.
121.                DB::commit();
122.
123.                return redirect('Pembeli/LihatAlamat/'.A
    uth::user()->id);
124.            }
125.            catch (\Throwable $th)
126.            {
127.                DB::rollback();
128.
129.                return $th;
130.            }
131.        }
132.    }
```

Kode Sumber 5.40 AlamatController

5.3.1.2. EtalaseController

Lapisan ini bertugas untuk melihat Semua etalase Pembeli. Kode Sumber 5.41 berikut merupakan implementasi dari lapisan kontrol EtalaseController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Pembeli;
4.
5.  use Carbon\Carbon;
6.  use App\Http\Controllers\Controller;
7.  use Illuminate\Http\Request;
8.  use App\Repositories\ProdukRepository;
9.  use App\Repositories\KategoriRepository;
10. use App\Repositories\BaseKategoriRepository;
```

```
11.  use App\Repositories\EtalaseRepository;
12.  use App\Repositories\KeranjangRepository;
13.  use App\Repositories\BarangKemasanRepository;
14.  use App\Repositories\IsiPaketRepository;
15.  use App\Repositories\BarangTanggalRepository;
16.  use App\Repositories\TanggalRepository;
17.
18.  use Illuminate\Support\Facades\Auth;
19.  use DB;
20.
21.  class EtalaseController extends Controller
22.  {
23.      protected $produkRepository,
24.                $kategoriRepository,
25.                $baseKategoriRepository,
26.                $etalaseRepository,
27.                $barangKemasanRepository,
28.                $isiPaketRepository,
29.                $keranjangRepository,
30.                $barangTanggalRepository,
31.                $tanggal;
32.
33.      public function __construct(BarangKemasanReposi
     tory $barangKemasanRepository, ProdukRepository $pr
     odukRepository, KategoriRepository $kategoriReposit
     ory, BaseKategoriRepository $baseKategoriRepository
     , EtalaseRepository $etalaseRepository, KeranjangRe
     pository $keranjangRepository, IsiPaketRepository $
     isiPaketRepository, BarangTanggalRepository $barang
     TanggalRepository, TanggalRepository $tanggal)
34.      {
35.          $this-
     >produkRepository = $produkRepository;
36.          $this-
     >kategoriRepository = $kategoriRepository;
```

```
37.             $this->baseKategoriRepository = $baseKategoriRepository;
38.             $this->etalaseRepository = $etalaseRepository;
39.             $this->keranjangRepository = $keranjangRepository;
40.             $this->barangKemasanRepository = $barangKemasanRepository;
41.             $this->isiPaketRepository = $isiPaketRepository;
42.             $this->barangTanggalRepository = $barangTanggalRepository;
43.             $this->tanggal      = $tanggal;
44.
45.             $this->middleware('auth');
46.             $this->middleware('pembeli');
47.         }
48.
49.         public function etalase()
50.         {
51.             return redirect('/Pembeli/Etalase/'.date('Y-m-d'));
52.         }
53.
54.         public function etalaseByKategori($NamaProduk)
55.         {
56.             return redirect('/Pembeli/Etalase/LihatProduks/'.$NamaProduk.'/'.date('Y-m-d'));
57.         }
58.
59.         public function etalaseBySubKategori($NamaKategori, $NamaSubKategori)
60.         {
```

```
61.             // dd("aaa");
62.             // dd('/Pembeli/Etalase/LihatProduk/'.$Nama
    Kategori.'/'.$NamaSubKategori.'/'.date('Y-m-d'));
63.             return redirect('/Pembeli/Etalase/LihatProd
    uk/'.$NamaKategori.'/'.$NamaSubKategori.'/'.date('Y
    -m-d'));
64.         }
65.
66.         public function cariEtalase($nama)
67.         {
68.             return redirect('/Pembeli/Etalase/CariProdu
    k/'.$nama.'/'.date('Y-m-d'));
69.
70.         }
71.
72.         public function detailProduk($id_barang)
73.         {
74.             $kategori = [
75.                 'sayur' => $this->kategoriRepository-
    >kategoriSayur(),
76.                 'buah' => $this->kategoriRepository-
    >kategoriBuah(),
77.                 'makanansehat' => $this-
    >kategoriRepository->kategoriMakananSehat(),
78.                 'minumansehat' => $this-
    >kategoriRepository->kategoriMinumanSehat(),
79.                 'beras' => $this->kategoriRepository-
    >kategoriBeras(),
80.                 'bahanolahan' => $this-
    >kategoriRepository->kategoriBahanOlahan(),
81.                 'berkebun' => $this-
    >kategoriRepository->kategoriBerkebun(),
82.                 'lainlain' => $this-
    >kategoriRepository->kategoriLainLain()
83.             ];
84.
```

```
85.             $get_barang = $this->produkRepository->findById($id_barang);
86.
87.             if($get_barang->jenis == 'Timbang')
88.             {
89.                 $barang_kemasan = $this->barangKemasanRepository->findByIdBarang($id_barang);
90.
91.                 $data = [
92.                     'barang' => $get_barang,
93.                     'kategori' => $this->kategoriRepository->array(),
94.                     'baseKategori' => $this->baseKategoriRepository->all()
95.                 ];
96.
97.                 return view('pembeli.detail-produk-timbang')
98.                     ->with(compact('data'))
99.                     ->with(compact('kategori'));
100.            }
101.
102.            else if($get_barang->jenis == 'Kemas')
103.            {
104.                $barang_kemasan = $this->barangKemasanRepository->findByIdBarang($id_barang);
105.
106.                $data = [
107.                    'barang' => $get_barang,
108.                    'kemasan' => $barang_kemasan,
109.                    'kategori' => $this->kategoriRepository->array(),
110.                    'baseKategori' => $this->baseKategoriRepository->all()
111.                ];
```

```
112.
113.                 return view('pembeli.detail-produk-
    kemas')
114.                     ->with(compact('data'))
115.                     ->with(compact('kategori'));
116.             }
117.
118.             else if($get_barang->jenis == 'Paket')
119.             {
120.                 $data = [
121.                     'barang' => $get_barang,
122.                     'isi_paket' => $this-
    >isiPaketRepository->read($get_barang->id),
123.                     'kategori' => $this-
    >kategoriRepository->array(),
124.                     'baseKategori' => $this-
    >baseKategoriRepository->all()
125.                 ];
126.
127.                 return view('pembeli.detail-produk-
    paket')
128.                     ->with(compact('data'))
129.                     ->with(compact('kategori'));
130.             }
131.         }
132.
133.         public function etalaseDate($date)
134.         {
135.             $tanggal= Carbon::now()->addDays(1)-
    >format('yy-m-d');
136.             $hapus=$this->tanggal-
    >hapusTanggal($tanggal);
137.             $this->barangTanggalRepository-
    >deleteByTanggalKurang($tanggal);
138.
139.             $data = [
```

```
140.                 'barang' => DB::select('call sp_get_barang_by_tanggal(?)',array($date)),
141.                 'barang_paket' => DB::select('call sp_get_barang_by_tanggal_and_jenis(?, ?)',array($date, 'Paket')),
142.                 'barang_timbang' => DB::select('call sp_get_barang_by_tanggal_and_jenis(?, ?)',array($date, 'Timbang')),
143.                 'barang_kemas' => DB::select('call sp_get_barang_by_tanggal_and_jenis(?, ?)',array($date, 'Kemas')),
144.                 'baseKategori' => $this->baseKategoriRepository->all(),
145.                 'tanggal_pengiriman' => $this->getTanggalPengiriman(),
146.                 'tanggal'=>$date
147.             ];
148.
149.             return view('pembeli.etalase')
150.                 ->with(compact('data'));
151.         }
152.         public function etalaseByKategoriDate($NamaProduk, $date)
153.         {
154.             switch($NamaProduk){
155.                 case 'Sayur':
156.                     $baseKategori =  $this->baseKategoriRepository->findByNamaKategori('Sayur');
157.                     $subkategori= $this->kategoriRepository->kategoriSayur();
158.                     break;
159.                 case 'Buah':
160.                     $baseKategori =  $this->baseKategoriRepository->findByNamaKategori('Buah');
```

```
161.                 $subkategori= $this-
      >kategoriRepository->kategoriBuah();
162.                 break;
163.             case 'Makanan Sehat':
164.                 $baseKategori =  $this-
      >baseKategoriRepository-
      >findByNamaKategori('Makanan Sehat');
165.                 $subkategori = $this-
      >kategoriRepository->kategoriMakananSehat();
166.
167.                 break;
168.             case 'Minuman Sehat':
169.                 $baseKategori =  $this-
      >baseKategoriRepository-
      >findByNamaKategori('Minuman Sehat');
170.                 $subkategori = $this-
      >kategoriRepository->kategoriMinumanSehat();
171.
172.                 break;
173.             case 'Daging dan Telor':
174.                 $baseKategori =  $this-
      >baseKategoriRepository-
      >findByNamaKategori('Daging dan Telor');
175.                 $subkategori = $this-
      >kategoriRepository->kategoriBahanOlahan();
176.                 break;
177.             case 'Beras':
178.                 $baseKategori =  $this-
      >baseKategoriRepository-
      >findByNamaKategori('Beras');
179.                 $subkategori = $this-
      >kategoriRepository->kategoriBeras();
180.
181.                 break;
182.             case 'Berkebun':
```

```
183.                 $baseKategori =  $this->baseKategoriRepository->findByNamaKategori('Berkebun');
184.                 $subkategori = $this->kategoriRepository->kategoriBerkebun();
185.
186.                 break;
187.             case 'Lain - Lain':
188.                 $baseKategori =  $this->baseKategoriRepository->findByNamaKategori('Lain - Lain');
189.                 $subkategori = $this->kategoriRepository->kategoriLainLain();
190.                 break;
191.         }
192.
193.         $kategori = [
194.             'sayur' => $this->kategoriRepository->kategoriSayur(),
195.             'buah' => $this->kategoriRepository->kategoriBuah(),
196.             'makanansehat' => $this->kategoriRepository->kategoriMakananSehat(),
197.             'minumansehat' => $this->kategoriRepository->kategoriMinumanSehat(),
198.             'beras' => $this->kategoriRepository->kategoriBeras(),
199.             'bahanolahan' => $this->kategoriRepository->kategoriBahanOlahan(),
200.             'berkebun' => $this->kategoriRepository->kategoriBerkebun(),
201.             'lainlain' => $this->kategoriRepository->kategoriLainLain()
202.         ];
203.
204.         $datas = [];
```

```
205.          $barang=DB::select('call sp_get_barang_by_t
      anggal_and_kategori(?, ?)',array($date, $baseKatego
      ri->id));
206.
207.          foreach($subkategori as $key => $bar){
208.              $datas[$key]['subkategori'] = $bar;
209.              $datas[$key]['barang'] = DB::select('ca
      ll sp_get_sub_kategor_by_barang_and_kateg(?, ?, ?)'
      ,array($baseKategori->id, $bar->id, $date));
210.
211.          }
212.
213.
214.          $data = [
215.              'barang' => collect($datas),
216.              'kategori' => $this-
      >kategoriRepository->array(),
217.              'baseKategori' => $this-
      >baseKategoriRepository->all(),
218.              'nama_produk' => $baseKategori-
      >kategori,
219.              'tanggal_pengiriman' => $this-
      >getTanggalPengiriman(),
220.              'tanggal'=>$date,
221.              'flag'=>0
222.          ];
223.
224.
225.
226.          return view('pembeli.etalase-
      subkategori')
227.              ->with(compact('data'))
228.              ->with(compact('kategori'));
229.      }
230.      public function etalaseBySubKategoriDate($NamaK
      ategori, $NamaSubKategori, $tanggal)
231.      {
```

```
232.          $kategori = [
233.              'sayur' => $this->kategoriRepository->kategoriSayur(),
234.              'buah' => $this->kategoriRepository->kategoriBuah(),
235.              'makanansehat' => $this->kategoriRepository->kategoriMakananSehat(),
236.              'minumansehat' => $this->kategoriRepository->kategoriMinumanSehat(),
237.              'beras' => $this->kategoriRepository->kategoriBeras(),
238.              'bahanolahan' => $this->kategoriRepository->kategoriBahanOlahan(),
239.              'berkebun' => $this->kategoriRepository->kategoriBerkebun(),
240.              'lainlain' => $this->kategoriRepository->kategoriLainLain()
241.          ];
242.
243.          $getProduk = $this->produkRepository->findByNamaSubKategori($NamaSubKategori, $tanggal);
244.
245.          foreach ($getProduk as $key => $value)
246.          {
247.              $getProduk[$key] = (object) $value;
248.          }
249.
250.          $data = [
251.              'barang' => $getProduk,
252.              'kategori' => $this->kategoriRepository->array(),
253.              'baseKategori' => $this->baseKategoriRepository->all(),
254.              'nama_subproduk' => $NamaSubKategori,
255.              'nama_kategori'=>$NamaKategori,
```

```
256.                 'tanggal_pengiriman' => $this->getTanggalPengiriman(),
257.                 'tanggal'=>$tanggal
258.             ];
259.
260.
261.             return view('pembeli.etalase-kategori')
262.                 ->with(compact('data'))
263.                 ->with(compact('kategori'));
264.         }
265.
266.         public function cariEtalaseDate($nama, $date)
267.         {
268.             $kategori = [
269.                 'sayur' => $this->kategoriRepository->kategoriSayur(),
270.                 'buah' => $this->kategoriRepository->kategoriBuah(),
271.                 'makanansehat' => $this->kategoriRepository->kategoriMakananSehat(),
272.                 'minumansehat' => $this->kategoriRepository->kategoriMinumanSehat(),
273.                 'beras' => $this->kategoriRepository->kategoriBeras(),
274.                 'bahanolahan' => $this->kategoriRepository->kategoriBahanOlahan(),
275.                 'berkebun' => $this->kategoriRepository->kategoriBerkebun(),
276.                 'lainlain' => $this->kategoriRepository->kategoriLainLain()
277.             ];
278.
279.             $data = [
280.                 'pencarian_produk' => $nama,
281.                 'barang' => DB::select('call sp_get_barang_by_tanggal_and_search(?, ?)',array($date, $nama)),
```

```
282.                 'kategori' => $this->kategoriRepository->array(),
283.                 'baseKategori' => $this->baseKategoriRepository->all(),
284.                 'tanggal_pengiriman' => $this->getTanggalPengiriman(),
285.                 'tanggal'=>$date
286.             ];
287.
288.             return view('pembeli.cariproduk')
289.                 ->with(compact('data'))
290.                 ->with(compact('kategori'));
291.         }
292.
293.         public function getTanggalPengiriman()
294.         {
295.
296.             $getListHari=$this->tanggal->get_tanggal();
297.
298.             if(count($getListHari)>0){
299.             for($a=0;$a<sizeof($getListHari);$a++)
300.             {
301.                 $tgl=$getListHari[$a]['tanggal'];
302.                 // string
303.                 $finalListHari[$a]['tanggal'] = Carbon::parse($tgl)->format('D, d F Y');
304.                 //tanggal value
305.                 $finalListHari[$a]['tanggal_value'] = Carbon::parse($tgl)->format('Y-m-d');
306.
307.             }
308.             }
309.             else{
310.                 $finalListHari[0]['tanggal'] = Carbon::now()->format('D, d F Y');
311.                 //tanggal value
```

```
312.                 $finalListHari[0]['tanggal_value'] = Ca
      rbon::now()->format('Y-m-d');
313.             }
314.
315.             // dd($finalListHari);
316.             foreach ($finalListHari as $key => $value)

317.             {
318.                 $finalListHari[$key] = (object) $value;

319.             }
320.
321.             // dd($finalListHari);
322.
323.             return $finalListHari;
324.         }
325.     }
```

Kode Sumber 5.41 EtalaseController

5.3.1.3. HomeController

Lapisan ini bertugas untuk melihat Semua Home Pembeli. Kode Sumber 5.42 berikut merupakan implementasi dari lapisan kontrol EtalaseController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Pembeli;
4.
5.  use App\Http\Controllers\Controller;
6.  use Illuminate\Http\Request;
7.
8.  class HomeController extends Controller
9.  {
10.     public function __construct()
11.     {
```

```
12.             $this->middleware('auth');
13.             $this->middleware('pembeli');
14.         }
15.         public function index()
16.         {
17.             return redirect('/Pembeli/Etalase');
18.         }
19.     }
```

Kode Sumber 5.42 HomeController

5.3.1.4. KeranjangController

Lapisan ini bertugas untuk melihat Semua Keranjang Pembeli. Kode Sumber 5.43 berikut merupakan implementasi dari lapisan kontrol KeranjangController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Pembeli;
4.
5.  use Carbon\Carbon;
6.  use App\Http\Controllers\Controller;
7.  use Illuminate\Http\Request;
8.  use Illuminate\Support\Facades\DB;
9.  use Webpatser\Uuid\Uuid;
10. use Illuminate\Support\Facades\Auth;
11.
12. use App\Repositories\BarangKemasanRepository;
13. use App\Repositories\KeranjangRepository;
14. use App\Repositories\DetailKeranjangRepository;
15. use App\Repositories\BobotKemasanRepository;
16. use App\Repositories\ResepRepository;
17. use App\Repositories\IsiResepRepository;
18. use App\Repositories\ProdukRepository;
19. use App\Repositories\IsiPaketRepository;
20. use App\Repositories\HariPengirimanRepository;
```

```
21.  use App\Repositories\TanggalRepository;
22.
23.
24.  use App\Models\Detail_keranjang as DetailKeranjang;
25.
26.
27.  class KeranjangController extends Controller
28.  {
29.      protected $keranjang;
30.      protected $detailKeranjang;
31.      protected $bobotKemasan;
32.      protected $resep;
33.      protected $isiResep;
34.      protected $barang;
35.      protected $isiPaket;
36.      protected $hariPengiriman;
37.      protected $barangKemasan;
38.      protected $tanggal;
39.
40.      public function __construct(BarangKemasanReposi
     tory $barangKemasan, HariPengirimanRepository $hari
     Pengiriman ,KeranjangRepository $keranjang, DetailK
     eranjangRepository $detailKeranjang, BobotKemasanRe
     pository $bobotKemasan, ResepRepository $resep, Isi
     ResepRepository $isiResep, ProdukRepository $barang
     , IsiPaketRepository $isiPaket, TanggalRepository $
     tanggal)
41.      {
42.          $this->middleware('auth');
43.          $this->middleware('pembeli');
44.
45.          $this->keranjang         = $keranjang;
46.          $this-
     >detailKeranjang   = $detailKeranjang;
47.          $this->bobotKemasan      = $bobotKemasan;
48.          $this->resep             = $resep;
```

```
49.             $this->isiResep          = $isiResep;
50.             $this->barang            = $barang;
51.             $this->isiPaket          = $isiPaket;
52.             $this-
    >hariPengiriman    = $hariPengiriman;
53.             $this-
    >barangKemasan     = $barangKemasan;
54.             $this->tanggal     = $tanggal;
55.         }
56.
57.         public function tambahItemKeranjang($idKeranjan
    g,Request $request)
58.         {
59.             // dd($request->all());
60.             try
61.             {
62.                 DB::beginTransaction();
63.
64.                 $this->detailKeranjang-
    >tambahDetailKeranjang($request->all());
65.
66.                 DB::commit();
67.
68.                 return 'ok';
69.             }
70.             catch (\Throwable $th)
71.             {
72.                 DB::rollback();
73.
74.                 return 'failed';
75.             }
76.         }
77.
78.         public function ubahItemKeranjang($id, Request
    $request)
79.         {
80.             try
```

```
81.             {
82.                 DB::beginTransaction();
83.
84.                 $this->detailKeranjang-
    >updateDetailKeranjang($id, $request->all());
85.
86.                 DB::commit();
87.
88.                 return 1;
89.             }
90.             catch (\Throwable $th)
91.             {
92.                 DB::rollback();
93.
94.                 return 0;
95.             }
96.         }
97.
98.         public function hapusItemKeranjang($id)
99.         {
100.            try
101.            {
102.                DB::beginTransaction();
103.
104.                $this->detailKeranjang-
    >hapusDetailKeranjang($id);
105.
106.                DB::commit();
107.
108.                return 'ok';
109.            }
110.            catch (\Throwable $th)
111.            {
112.                DB::rollback();
113.
114.                return 'failed';
115.            }
```

```
116.        }
117.
118.        public function getBobotKemasan()
119.        {
120.            return $this->bobotKemasan->all();
121.        }
122.
123.        public function showInputItemKeranjang($id)
124.        {
125.            $cekBarang          = $this->barang-
    >findById($id);
126.
127.            if($cekBarang->jenis == 'Kemas')
128.            {
129.
130.                $data = [
131.                    'barang' => $cekBarang,
132.                    // 'bobot_kemasan' => $this-
    >bobotKemasan->all()
133.                    'bobot_kemasan' => $this-
    >barangKemasan->findByIdBarang($id)
134.                ];
135.
136.                return view('pembeli.components.input-
    item-kemas')
137.                    ->with(compact('data'))
138.                    ->render();
139.            }
140.            else if($cekBarang->jenis == 'Paket')
141.            {
142.                $getIsiPaket = $this->isiPaket-
    >findByIdBarang($id);
143.
144.                foreach($getIsiPaket as $isiPaket)
145.                {
146.                    $getBarang              = $this-
    >barang->findById($isiPaket->id_barang);
```

```
147.
148.                     $isiPaket->nama_barang = $getBarang->nama;
149.                     $isiPaket->satuan      = $getBarang->satuan;
150.                 }
151.
152.                 $data = [
153.                     'barang'    => $cekBarang,
154.                     'isiPaket'  => $getIsiPaket
155.                 ];
156.
157.                 return view('pembeli.components.input-item-paket')
158.                     ->with(compact('data'))
159.                     ->render();
160.             }
161.             else if($cekBarang->jenis == 'Timbang')
162.             {
163.                 $data = [
164.                     'barang' => $cekBarang,
165.                 ];
166.
167.                 return view('pembeli.components.input-item-timbang')
168.                     ->with(compact('data'))
169.                     ->render();
170.             }
171.         }
172.
173.         public function showUbahItemKeranjang($id)
174.         {
175.
176.             $getDetailKeranjang = $this->detailKeranjang->findById($id);
177.             $cekBarang = $this->barang->findById($getDetailKeranjang->id_barang);
```

```
178.
179.           if($cekBarang->jenis == 'Kemas')
180.           {
181.
182.               $data = [
183.                   'barang' => $cekBarang,
184.                   'bobot_kemasan' => $this-
      >barangKemasan->findByIdBarang($getDetailKeranjang-
      >id_barang),
185.                   'isi_keranjang' => $getDetailKeranj
      ang
186.               ];
187.
188.               return view('pembeli.components.ubah-
      item-kemas')
189.                   ->with(compact('data'))
190.                   ->render();
191.           }
192.           else if($cekBarang->jenis == 'Paket')
193.           {
194.               $getIsiPaket = $this->isiPaket-
      >findByIdBarang($getDetailKeranjang->id_barang);
195.
196.               foreach($getIsiPaket as $isiPaket)
197.               {
198.                   $getBarang             = $this-
      >barang->findById($isiPaket->id_barang);
199.
200.                   $isiPaket-
      >nama_barang = $getBarang->nama;
201.                   $isiPaket-
      >satuan      = $getBarang->satuan;
202.               }
203.
204.               $data = [
205.                   'barang'    => $cekBarang,
206.                   'isiPaket'  => $getIsiPaket,
```

```
                    'isi_keranjang' => $getDetailKeranjang
                ];

                return view('pembeli.components.ubah-item-paket')
                    ->with(compact('data'))
                    ->render();
            }
            else if($cekBarang->jenis == 'Timbang')
            {
                $data = [
                    'barang' => $cekBarang,
                    'isi_keranjang' => $getDetailKeranjang
                ];

                return view('pembeli.components.ubah-item-timbang')
                    ->with(compact('data'))
                    ->render();
            }
        }

        public function submitInputItemKeranjang($id, Request $request)
        {

            try
            {
                DB::beginTransaction();

                if($this->keranjang->findKeranjang($request->tanggal) == null)
                {
                    $this->keranjang->tambahKeranjang($request->tanggal);
```

```
237.                 }
238.
239.                 $getIdKeranjang = $this->keranjang-
    >getIDKeranjang($request->tanggal);
240.
241.                 switch($this->barang->findById($id)-
    >jenis)
242.                 {
243.                     case 'Paket':
244.                         if($this->barang-
    >findById($id)->diskon>100){
245.                             $harga_diskon=($this-
    >barang->findById($id)->harga_jual-$this->barang-
    >findById($id)->diskon) * $request->total_order;
246.                         }
247.                         else{
248.                             $harga_diskon=($this-
    >barang->findById($id)->harga_jual-($this->barang-
    >findById($id)->harga_jual*($this->barang-
    >findById($id)->diskon/100))) * $request-
    >total_order;
249.                         }
250.                         if($this->detailKeranjang-
    >findByIdKeranjangIdBarang($getIdKeranjang,$id) ==
    null)
251.                         {
252.                             $data = [
253.                                 'id'           => Uuid:
    :generate(),
254.                                 'id_keranjang' => $getI
    dKeranjang,
255.                                 'id_barang'    => $id,

256.                                 'volume'       => $requ
    est->total_order,
```

```
257.                                 'harga'         => $this
->barang->findById($id)->harga_jual * $request-
>total_order,
258.                                 'harga_diskon' => $harg
a_diskon
259.                             ];
260.
261.                             $this->detailKeranjang-
>tambahDetailKeranjang($data);
262.
263.                             DB::commit();
264.                             return 1;
265.                         }
266.                         else
267.                         {
268.                             $currentVolume = $this-
>detailKeranjang-
>findByIdKeranjangIdBarang($getIdKeranjang,$id)-
>volume;
269.                             $currentHarga  = $this-
>detailKeranjang-
>findByIdKeranjangIdBarang($getIdKeranjang,$id)-
>harga;
270.                             $currentHargaDiskon  = $thi
s->detailKeranjang-
>findByIdKeranjangIdBarang($getIdKeranjang,$id)-
>harga_diskon;
271.
272.                             $data = [
273.                                 'volume'        => $requ
est->total_order + $currentVolume,
274.                                 'harga'         => ($thi
s->barang->findById($id)->harga_jual * $request-
>total_order) + $currentHarga,
275.                                 'harga_diskon' => $harg
a_diskon+$currentHargaDiskon,
276.                             ];
```

```
277.
278.                             $this->detailKeranjang-
    >updateDetailKeranjang($id, $data);
279.
280.                             DB::commit();
281.                             return 1;
282.                         }
283.
284.                     case 'Timbang':
285.                         if($this->barang-
    >findById($id)->diskon>100){
286.                             $harga_diskon=(($this-
    >barang->findById($id)->harga_jual/10)-($this-
    >barang->findById($id)->diskon/10)) * ($request-
    >total_order/100);
287.                         }
288.                         else{
289.                             $harga_diskon=(($this-
    >barang->findById($id)->harga_jual/10)-(($this-
    >barang->findById($id)->harga_jual/10)*($this-
    >barang->findById($id)->diskon/100))) * (($request-
    >total_order)/100);
290.                         }
291.
292.                         if($this->detailKeranjang-
    >findByIdKeranjangIdBarang($getIdKeranjang,$id) ==
    null)
293.                         {
294.                             $data = [
295.                                 'id'            => Uuid:
    :generate(),
296.                                 'id_keranjang' => $getI
    dKeranjang,
297.                                 'id_barang'     => $id,

298.                                 'volume'        => $requ
    est->total_order,
```

```
299.                                 'harga'         => ($thi
      s->barang->findById($id)-
      >harga_jual/10) * (($request->total_order)/100),
300.                                 'harga_diskon' => $harg
      a_diskon
301.                             ];
302.
303.                             $this->detailKeranjang-
      >tambahDetailKeranjang($data);
304.
305.                             DB::commit();
306.                             return 1;
307.                         }
308.                         else
309.                         {
310.                             $currentVolume = $this-
      >detailKeranjang-
      >findByIdKeranjangIdBarang($getIdKeranjang,$id)-
      >volume;
311.                             $currentHarga  = $this-
      >detailKeranjang-
      >findByIdKeranjangIdBarang($getIdKeranjang,$id)-
      >harga;
312.                             $currentHargaDiskon  = $thi
      s->detailKeranjang-
      >findByIdKeranjangIdBarang($getIdKeranjang,$id)-
      >harga_diskon;
313.
314.                             $data = [
315.                                 'volume'        => $requ
      est->total_order + $currentVolume,
316.                                 'harga'         => ($thi
      s->barang->findById($id)-
      >harga_jual/10) * (($request-
      >total_order)/100) + $currentHarga,
317.                                 'harga_diskon' => $harg
      a_diskon + $currentHargaDiskon
```

```
318.                             ];
319.
320.                             $this->detailKeranjang->updateDetailKeranjang($id, $data);
321.
322.                             DB::commit();
323.                             return 1;
324.                         }
325.
326.                     case 'Kemas':
327.                         if($this->barang->findById($id)->diskon>100){
328.                             $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-($this->barang->findById($id)->diskon/10)) * (($request->total_order*$request->volume_order)/100);
329.                         }
330.                         else{
331.                             $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-(($this->barang->findById($id)->harga_jual/10)*($this->barang->findById($id)->diskon/100))) * (($request->total_order*$request->volume_order)/100);
332.                         }
333.                         if($this->detailKeranjang->findByIdBarangBobotKemasan($getIdKeranjang, $id, (int) $request->volume_order) == null)
334.                         {
335.                             $data = [
336.                                 'id'           => Uuid::generate(),
337.                                 'id_keranjang' => $getIdKeranjang,
338.                                 'id_barang'    => $id,
339.                                 'volume'       => $request->total_order,
```

```
340.                                'bobot_kemasan'=> $request->volume_order,
341.                                'harga'         => ($this->barang->findById($id)->harga_jual/10) * (($request->total_order*$request->volume_order)/100),
342.                                'harga_diskon' => $harga_diskon
343.                            ];
344.
345.                            $this->detailKeranjang->tambahDetailKeranjang($data);
346.
347.                            DB::commit();
348.                            return 1;
349.                        }
350.                        else
351.                        {
352.                            $currentVolume = $this->detailKeranjang->findByIdBarangBobotKemasan($getIdKeranjang,$id,(int) $request->volume_order)->volume;
353.                            $currentHarga  = $this->detailKeranjang->findByIdBarangBobotKemasan($getIdKeranjang,$id,(int) $request->volume_order)->harga;
354.                            $currentHargaDiskon  = $this->detailKeranjang->findByIdBarangBobotKemasan($getIdKeranjang,$id,(int) $request->volume_order)->harga_diskon;
355.
356.                            $data = [
357.                                'volume'        => $request->total_order + $currentVolume,
358.                                'harga'         => (($this->barang->findById($id)-
```

```
>harga_jual/10) * (($request->total_order*$request->volume_order)/100)) + $currentHarga,
                                'harga_diskon' => $harga_diskon+$currentHargaDiskon
                            ];

                            $this->detailKeranjang->updateDetailKeranjangKemas($id,(int) $request->volume_order,$data);

                            DB::commit();
                            return 1;
                        }
                }
            }
            catch (\Throwable $th)
            {
                DB::rollback();
                return $th->getMessage();
            }
        }

        public function submitUbahItemKeranjang($id, Request $request)
        {
            try
            {
                DB::beginTransaction();

                if($this->keranjang->findKeranjang($request->tanggal) == null)
                {
                    $this->keranjang->tambahKeranjang($request->tanggal);
                }

```

```
387.                $getIdKeranjang = $this->keranjang->getIDKeranjang($request->tanggal);
388.
389.                switch($this->barang->findById($id)->jenis)
390.                {
391.                    case 'Paket':
392.                            if($this->barang->findById($id)->diskon>100){
393.                                $harga_diskon=($this->barang->findById($id)->harga_jual-$this->barang->findById($id)->diskon) * $request->total_order;
394.                            }
395.                            else{
396.                                $harga_diskon=($this->barang->findById($id)->harga_jual-($this->barang->findById($id)->harga_jual*($this->barang->findById($id)->diskon/100))) * $request->total_order;
397.                            }
398.                            $data = [
399.                                'id'            => Uuid::generate(),
400.                                'id_keranjang' => $getIdKeranjang,
401.                                'id_barang'     => $id,
402.                                'volume'        => $request->total_order,
403.                                'harga'         => $this->barang->findById($id)->harga_jual * $request->total_order,
404.                                'harga_diskon' => $harga_diskon,
405.                            ];
406.
```

```
407.                             $this->detailKeranjang->updateDetailKeranjang($id, $data);
408.
409.                             DB::commit();
410.                             return 1;
411.
412.                     case 'Timbang':
413.                             if($this->barang->findById($id)->diskon>100){
414.                                 $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-($this->barang->findById($id)->diskon/10)) * ($request->total_order/100);
415.                             }
416.                             else{
417.                                 $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-(($this->barang->findById($id)->harga_jual/10)*($this->barang->findById($id)->diskon/100))) * (($request->total_order)/100);
418.                             }
419.                             $data = [
420.                                 'id'            => Uuid::generate(),
421.                                 'id_keranjang' => $getIdKeranjang,
422.                                 'id_barang'     => $id,
423.                                 'volume'        => $request->total_order,
424.                                 'harga'         => ($this->barang->findById($id)->harga_jual/10) * (($request->total_order)/100),
425.                                 'harga_diskon' => $harga_diskon
426.                             ];
427.
```

```
428.                             $this->detailKeranjang->updateDetailKeranjang($id, $data);
429.
430.                             DB::commit();
431.                             return 1;
432.
433.                     case 'Kemas':
434.                             if($this->barang->findById($id)->diskon>100){
435.                                 $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-($this->barang->findById($id)->diskon/10)) * ($request->total_order/100);
436.                             }
437.                             else{
438.                                 $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-(($this->barang->findById($id)->harga_jual/10)*($this->barang->findById($id)->diskon/100))) * (($request->total_order)/100);
439.                             }
440.                             $data = [
441.                                 'id'            => Uuid::generate(),
442.                                 'id_keranjang' => $getIdKeranjang,
443.                                 'id_barang'     => $id,
444.                                 'volume'        => $request->total_order,
445.                                 'bobot_kemasan'=> $request->volume_order,
446.                                 'harga'         => ($this->barang->findById($id)->harga_jual/10) * (($request->total_order*$request->volume_order)/100),
```

```
                                'harga_diskon' => $harga_diskon
                            ];

                            $this->detailKeranjang->updateDetailKeranjang($id, $data);

                            DB::commit();
                            return 1;
                        }
                }
                catch (\Throwable $th)
                {
                    DB::rollback();
                    return $th->getMessage();
                }
            }

            public function lihatItemKeranjang($date)
            {
                if($this->keranjang->findKeranjang($date) == null){
                    $this->keranjang->tambahKeranjang($date);
                }

                $getIDKeranjang       = $this->keranjang->getIDKeranjang($date);
                $getDetailKeranjang   = $this->detailKeranjang->getDetailKeranjang($getIDKeranjang);
                $total=0;

                foreach($getDetailKeranjang as $detailKeranjang)
                {
```

```
475.                $getBarang = $this->barang-
    >findById($detailKeranjang->id_barang);
476.                if($getBarang->jenis == 'Kemas')
477.                {
478.                    $detailKeranjang-
    >nama       = $getBarang-
    >nama.' ('.$detailKeranjang-
    >bobot_kemasan.' '.'Gram'.')';
479.                    $detailKeranjang-
    >jenis      = $getBarang->jenis;
480.                    $detailKeranjang-
    >satuan     = 'Kemas';
481.                    $detailKeranjang-
    >bobot      = $getBarang->bobot;
482.                    $detailKeranjang-
    >volume     = $detailKeranjang->volume;
483.                }
484.                else if($getBarang-
    >jenis == 'Paket' || $getBarang-
    >jenis == 'Timbang')
485.                {
486.                    $detailKeranjang-
    >nama       = $getBarang->nama;
487.                    $detailKeranjang-
    >jenis      = $getBarang->jenis;
488.                    $detailKeranjang-
    >satuan     = $getBarang->satuan;
489.                    $detailKeranjang-
    >bobot      = $getBarang->bobot;
490.                }
491.                $total+= $detailKeranjang-
    >harga_diskon;
492.            }
493.
494.            $data = [
495.                'total'=>$total,
```

```
496.                 'detail_keranjang'  => $getDetailKeranj
    ang,
497.                 'tanggal_pengiriman' => $date
498.             ];
499.             // dd($data['total']);
500.
501.             return view('pembeli.components.lihat-
    keranjang')
502.                 ->with(compact('data'))
503.                 ->render();
504.         }
505.
506.         public function getTanggalPengiriman()
507.         {
508.
509.              $getListHari=$this->tanggal-
    >get_tanggal();
510.
511.             for($a=0;$a<sizeof($getListHari);$a++)
512.             {
513.                 $tgl=$getListHari[$a]['tanggal'];
514.                 // string
515.                 $finalListHari[$a]['tanggal'] = Carbon:
    :parse($tgl)->format('D, d F Y');
516.                 //tanggal value
517.                 $finalListHari[$a]['tanggal_value'] = C
    arbon::parse($tgl)->format('Y-m-d');
518.
519.                 // dd($finalListHari[$a]['tanggal_value
    ']);
520.             }
521.
522.
523.
524.             foreach ($finalListHari as $key => $value)

525.             {
```

```
526.                $finalListHari[$key] = (object) $value;

527.            }
528.
529.            // dd($finalListHari);
530.
531.            return $finalListHari;
532.        }
533.    }
```

Kode Sumber 5.43 KeranjangController

5.3.1.5. ProdukController

Lapisan ini bertugas untuk melakukan hal yg berhubungan dengan Produk di Pembeli. Kode Sumber 5.44 berikut merupakan implementasi dari lapisan kontrol ProdukController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Pembeli;
4.
5.  use App\Http\Controllers\Controller;
6.  use Illuminate\Http\Request;
7.
8.  use App\Repositories\BarangRepository;
9.  use App\Repositories\KeranjangRepository;
10. use App\Repositories\BaseKategoriRepository;
11. use App\Repositories\KategoriRepository;
12.
13. use DB;
14.
15. class ProdukController extends Controller
16. {
17.     protected $barang;
18.     protected $baseKategori;
19.     protected $kategori;
```

```
20.         protected $keranjang;
21.
22.         public function __construct(BarangRepository $b
    arang, BaseKategoriRepository $baseKategori, Katego
    riRepository $kategori, KeranjangRepository $keranj
    ang)
23.         {
24.             $this->middleware('auth');
25.             $this->middleware('pembeli');
26.
27.             $this->barang       = $barang;
28.             $this->baseKategori = $baseKategori;
29.             $this->kategori     = $kategori;
30.             $this->keranjang    = $keranjang;
31.         }
32.
33.
34.         private function showDetailBarangPaket($id)
35.         {
36.
37.         }
38.
39.         private function showDetailBarangResep($id)
40.         {
41.
42.         }
43.
44.         private function showDetailBarangBiasa($id)
45.         {
46.             $data = [
47.                 'detailBarang' => $this->barang-
    >showDetilBarang(),
48.                 'baseKategori' => $this->baseKategori-
    >getAllBaseKategori(),
49.                 'kategori' => $this->kategori-
    >getAll()
50.             ];
```

```
51.         }
52.
53.         public function showBarang()
54.         {
55.             $data = [
56.                 'barang'         => $this->barang-
    >getAllBarang(),
57.                 'baseKategori'   => $this->baseKategori-
    >getAllBaseKategori(),
58.                 'kategori'       => $this->kategori-
    >getAll()
59.             ];
60.
61.             if($this->keranjang-
    >findKeranjang() == null)
62.             {
63.                 $this->keranjang->tambahKeranjang();
64.             }
65.
66.             return view('pembeli.etalase')-
    >with(compact('data'));
67.         }
68.
69.         public function showBarangByBaseKategori($id)
70.         {
71.             $data = [
72.                 'barang' => $this->barang-
    >barangByKategori(),
73.                 'baseKategori' => $this->baseKategori-
    >getAllBaseKategori(),
74.                 'kategori' => $this->kategori-
    >getAll()
75.             ];
76.         }
77.
78.         public function percobaan()
79.         {
```

```
80.
81.         }
82.
83.     }
```

Kode Sumber 5.44 ProdukController

5.3.1.6. ProfilController

Lapisan ini bertugas untuk mengatur Profil Pembeli. Kode Sumber 5.45 berikut merupakan implementasi dari lapisan kontrol ProfilController:

```
1.      <?php
2.
3.      namespace App\Http\Controllers\Pembeli;
4.
5.      use App\Http\Controllers\Controller;
6.      use Illuminate\Http\Request;
7.      use Auth;
8.      use DB;
9.
10.     use App\Repositories\KategoriRepository;
11.     use App\Repositories\ProdukRepository;
12.     use App\Repositories\BaseKategoriRepository;
13.     use App\Repositories\UserRepository;
14.
15.     class ProfilController extends Controller
16.     {
17.         protected $kategoriRepository, $produkRepositor
        y, $baseKategoriRepository, $userRepository;
18.
19.         public function __construct(KategoriRepository
        $kategoriRepository, ProdukRepository $produkReposi
        tory, BaseKategoriRepository $baseKategoriRepositor
        y, UserRepository $userRepository)
20.         {
```

```
21.             $this->kategoriRepository = $kategoriRepository;
22.             $this->produkRepository = $produkRepository;
23.             $this->baseKategoriRepository = $baseKategoriRepository;

24.             $this->userRepository=$userRepository;
25.
26.             $this->middleware('auth');
27.             $this->middleware('pembeli');
28.         }
29.
30.         public function viewProfile()
31.         {
32.             $kategori = [
33.                 'sayur' => $this->kategoriRepository->kategoriSayur(),
34.                 'buah' => $this->kategoriRepository->kategoriBuah(),
35.                 'makanansehat' => $this->kategoriRepository->kategoriMakananSehat(),
36.                 'minumansehat' => $this->kategoriRepository->kategoriMinumanSehat(),
37.                 'beras' => $this->kategoriRepository->kategoriBeras(),
38.                 'bahanolahan' => $this->kategoriRepository->kategoriBahanOlahan(),
39.                 'berkebun' => $this->kategoriRepository->kategoriBerkebun(),
40.                 'lainlain' => $this->kategoriRepository->kategoriLainLain()
41.             ];
42.
43.             $data = [
44.                 'barang' => $this->produkRepository->showEtalase(),
```

```
45.                 'barang_paket' => $this->produkRepository->showByJenis('Paket'),
46.                 'barang_timbang' => $this->produkRepository->showByJenis('Timbang'),
47.                 'barang_kemas' => $this->produkRepository->showByJenis('Kemas'),
48.                 'kategori' => $this->kategoriRepository->array(),
49.                 'baseKategori' => $this->baseKategoriRepository->all(),
50.                 'user' => Auth::user()
51.             ];
52.
53.             return view('pembeli.profil')
54.                 ->with(compact('data'))
55.                 ->with(compact('kategori'));
56.         }
57.
58.         public function editProfile(){
59.             $data=[
60.                 'user'=>Auth::user()
61.             ];
62.             return view('pembeli.ubah-profil')->with(compact('data'));
63.         }
64.
65.         public function _editProfile(Request $request){

66.             try
67.             {
68.                 DB::beginTransaction();
69.
70.                 $this->userRepository->update($request->all());
71.
72.                 DB::commit();
73.
```

```
                return redirect('Pembeli/Profil/');
            }
            catch (\Throwable $th)
            {
                DB::rollback();

                return $th;
            }
        }
    }
```

Kode Sumber 5.45 ProfilController

5.3.1.7. TransaksiController

Lapisan ini bertugas untuk mengatur transaksi Pembeli. Kode Sumber 5.46 berikut merupakan implementasi dari lapisan kontrol TransaksiController:

```
<?php

namespace App\Http\Controllers\Pembeli;

use Auth;
use Carbon\Carbon;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Repositories\TransaksiRepository;
use App\Repositories\DetailTransaksiRepository;
use App\Repositories\KeranjangRepository;
use App\Repositories\DetailKeranjangRepository;
use App\Repositories\AlamatRepository;
use App\Repositories\ProdukRepository;
use App\Repositories\IsiPaketRepository;
use App\Repositories\UserRepository;
use Uuid;
use DB;
```

```
19.
20.     class TransaksiController extends Controller
21.     {
22.         protected $barang;
23.         protected $transaksi;
24.         protected $keranjang;
25.         protected $detailTransaksi;
26.         protected $detailKeranjang;
27.         protected $alamat;
28.         protected $isiPaket;
29.         protected $user;
30.
31.         public function __construct(ProdukRepository $b
        arang, TransaksiRepository $transaksi, KeranjangRep
        ository $keranjang, DetailTransaksiRepository $deta
        ilTransaksi, DetailKeranjangRepository $detailKeran
        jang, AlamatRepository $alamat, IsiPaketRepository
        $isiPaket, UserRepository $user)
32.         {
33.             $this->middleware('auth');
34.             $this->middleware('pembeli');
35.
36.             $this->transaksi        = $transaksi;
37.             $this-
        >detailTransaksi  = $detailTransaksi;
38.             $this->keranjang        = $keranjang;
39.             $this-
        >detailKeranjang  = $detailKeranjang;
40.             $this->alamat           = $alamat;
41.             $this->barang           = $barang;
42.             $this->isiPaket         = $isiPaket;
43.             $this->user             = $user;
44.         }
45.
46.         public function viewCheckout(Request $request)

47.         {
```

```
48.             $alamat=$this->alamat-
    >getAllAlamatByUser(Auth::user()->id);
49.             if(count($alamat)==0){
50.                 return redirect('/Pembeli/TambahAlamat/
    ');
51.             }
52.             $getKeranjang           = $this->keranjang-
    >getIDKeranjang($request->tanggal_pre_order);
53.             $getDetailKeranjang     = $this-
    >detailKeranjang-
    >getDetailKeranjang($getKeranjang);
54.             $totalHarga             = 0;
55.             $detailPaketKeranjang   = array();
56.             $totalHargaPaket            = 0;
57.             $totalHargaKemas            = 0;
58.             $totalHargaTimbang            = 0;
59.
60.             $dataInput = [
61.                 'tanggal_pre_order' => $request-
    >tanggal_pre_order
62.             ];
63.
64.             $this->keranjang-
    >updateKeranjang($getKeranjang, $dataInput);
65.
66.
67.             foreach($getDetailKeranjang as $detailKeran
    jang)
68.             {
69.                 $getBarang                      = $this
    ->barang->findById($detailKeranjang->id_barang);
70.
71.                 if($getBarang->jenis == 'Kemas')
72.                 {
73.                     $detailKeranjang-
    >nama          = $detailKeranjang-
    >bobot_kemasan.' '.'Gram'.' '.$getBarang->nama;
```

```
74.                     $detailKeranjang->jenis          = $getBarang->jenis;
75.                     $detailKeranjang->satuan         = $getBarang->satuan;
76.                     $detailKeranjang->bobot          = $getBarang->bobot;
77.                     $detailKeranjang->harga_satuan   = $getBarang->harga_jual;
78.                     $detailKeranjang->volume         = $detailKeranjang->volume * ($getBarang->bobot/10);
79.                     $detailKeranjang->diskon         = $getBarang->diskon;
80.                     $detailKeranjang->jenis_diskon   = "Potongan Persen";
81.
82.                     $totalHarga = $totalHarga + $detailKeranjang->harga_diskon;
83.                 }
84.                 else if($getBarang->jenis == 'Paket')
85.                 {
86.                     $detailKeranjang->nama           = $getBarang->nama;
87.                     $detailKeranjang->jenis          = $getBarang->jenis;
88.                     $detailKeranjang->satuan         = $getBarang->satuan;
89.                     $detailKeranjang->bobot          = $getBarang->bobot;
90.                     $detailKeranjang->harga_satuan   = $getBarang->harga_jual;
91.                     $detailKeranjang->isPaket        = $getBarang->is_paket;
92.                     $detailKeranjang->diskon         = $getBarang->diskon;
93.                     $detailKeranjang->jenis_diskon   = $getBarang->jenis_diskon;
```

```
94.
95.                     $getIsiPaket = $this->isiPaket-
    >findByIdBarang($detailKeranjang->id_barang);
96.                     $getIsiPaket = array($getIsiPaket);
97.
98.                     for($a=0;$a<sizeof($getIsiPaket[0])
    ;$a++)
99.                     {
100.                        $getBarang = $this->barang-
    >findById($getIsiPaket[0][$a]->id_barang);
101.
102.                        $detailPaketKeranjang[$detailKe
    ranjang->id_barang][$a]['nama']   = $getBarang-
    >nama;
103.                        $detailPaketKeranjang[$detailKe
    ranjang-
    >id_barang][$a]['volume'] = $getIsiPaket[0][$a]-
    >volume;
104.                        $detailPaketKeranjang[$detailKe
    ranjang->id_barang][$a]['satuan'] = $getBarang-
    >satuan;
105.                    }
106.
107.                    $totalHarga = $totalHarga + $detail
    Keranjang->harga_diskon;
108.                }
109.                else if($getBarang-
    >jenis == 'Timbang')
110.                {
111.                    $detailKeranjang-
    >nama           = $getBarang->nama;
112.                    $detailKeranjang-
    >jenis          = $getBarang->jenis;
113.                    $detailKeranjang-
    >satuan         = $getBarang->satuan;
```

```
                    $detailKeranjang->bobot          = $getBarang->bobot;
                    $detailKeranjang->harga_satuan   = $getBarang->harga_jual;
                    $detailKeranjang->diskon         = $getBarang->diskon;
                    $detailKeranjang->jenis_diskon   = "Potongan Persen";

                    $totalHarga = $totalHarga + $detailKeranjang->harga_diskon;
                }
            // dd($totalHargaKemas);

            }
            $totalHarga = $totalHarga;
            if($this->transaksi->checkTransaksi($request->tanggal_pre_order) == null)
            {
                $data = [
                    'tanggal_pre_order'     => $request->tanggal_pre_order,
                    'isCheckout'            => 0,
                    'total_bayar'           => $totalHarga
                ];

                $this->transaksi->halamanCheckout($data);
            }

            else
            {
                $data = [
                    'tanggal_pre_order'     => $request->tanggal_pre_order,
```

```
                'total_bayar'           => $totalHarga
            ];

            $this->transaksi->updateHalamanCheckout($data);
        }

        $getAlamat = $this->alamat->getAllAlamatByUser(Auth::user()->id);

        $data = [
            'keranjang'             => $this->keranjang->getKeteranganKeranjang($request->tanggal_pre_order),
            'detail_keranjang'      => $getDetailKeranjang,
            'total'                 => $totalHarga,
            'detail_paket_keranjang' => $detailPaketKeranjang,
            'alamat'                => $getAlamat
        ];

        // dd($data['keranjang']);

        return view('pembeli.checkout')->with(compact('data'));
    }

    public function submitCheckout(Request $request)
    {
        // dd($request->alamat);
        $getKeranjang           = $this->keranjang->getIDKeranjang($request->tanggal_pre_order);
```

```
165.
166.          $getDetailKeranjang     = $this->detailKeranjang->getDetailKeranjang($getKeranjang);
167.          $getTransaksi           = $this->transaksi->checkTransaksi($request->tanggal_pre_order);
168.          // dd($request->tanggal_pre_order);
169.          $nomor=$getTransaksi->nomor_invoice;
170.
171.          foreach($getDetailKeranjang as $detail)
172.          {
173.              // dd($detail->bobot_kemasan);
174.              $data = [
175.                  'id_barang'     => $detail->id_barang,
176.                  'harga'         => $detail->harga,
177.                  'harga_diskon'  => $detail->harga_diskon,
178.                  'volume'        => $detail->volume,
179.                  'id_transaksi'  => $getTransaksi->id,
180.                  'bobot_kemasan' => $detail->bobot_kemasan
181.              ];
182.
183.              $this->detailTransaksi->inputDetailTransaksi($data);
184.          }
185.
186.
187.          $data = [
188.              'keterangan' => $request->keterangan,
189.              'id_alamat'  => $request->alamat
190.          ];
191.
```

```
192.            $this->transaksi-
        >purchaseCheckout($data, $request-
        >tanggal_pre_order);
193.
194.            $this->keranjang-
        >hapusKeranjang($getKeranjang);
195.            $this->detailKeranjang-
        >hapusDetailKeranjangByIdKeranjang($getKeranjang);

196.
197.            return view('pembeli.transaksi-sukses')-
        >with(compact('nomor'));
198.        }
199.
200.        public function showTransaksi($tipe){
201.            $alltype=[
202.                [
203.                    'status'=>"BelumDibayar",
204.                    'status2' => "Belum Dibayar"
205.                ],
206.                [
207.                    'status'=>"Proses",
208.                    'status2' => "Dalam Proses"
209.                ],
210.                [
211.                    'status'=>"Selesai",
212.                    'status2' => "Selesai"
213.                ],
214.                [
215.                    'status'=>"Dibatalkan",
216.                    'status2' => "Dibatalkan"
217.                ],
218.            ];
219.            if($tipe=="BelumDibayar"){
220.                $tipe2="Belum Dibayar";
```

```
221.                 $transaksi = $this->transaksi-
      >getAllTransaksiByUserAndNotPay(Auth::user()-
      >id);
222.             }
223.             else if($tipe=="Proses"){
224.                 $tipe2="Dalam Proses";
225.                 $transaksi = $this->transaksi-
      >getAllTransaksiByUserAndInProcess(Auth::user()-
      >id);
226.             }
227.             else if($tipe=="Selesai"){
228.                 $tipe2="Selesai";
229.                 $transaksi = $this->transaksi-
      >getAllTransaksiByUserAndIsFinish(Auth::user()-
      >id);
230.             }
231.             else if($tipe=="Dibatalkan"){
232.                 $tipe2="Dibatalkan";
233.                 $transaksi = $this->transaksi-
      >getAllTransaksiByUserAndIsCancelled(Auth::user()-
      >id);
234.             }
235.             else{
236.                 return redirect('/');
237.             }
238.             $data = [
239.                 'alltype'=>$alltype,
240.                 'tipe'=>$tipe,
241.                 'tipe2'=>$tipe2,
242.                 'transaksi'=>$transaksi,
243.                 'rekening'=>$this->user->getVeggo()
244.             ];
245.
246.             return view('pembeli.transaksi')-
      >with('data',$data);
247.         }
248.
```

```
249.      public function _filterTanggal(Request $request
      ){
250.          session(['tanggal_pre_order' => $request-
      >input('tanggal')]);
251.          return redirect('/Pembeli/Transaksi');
252.      }
253.
254.      public function kirimBukti($id_transaksi,Reques
      t $request)
255.      {
256.
257.          // dd('masuk bang');
258.          $file = $request->file('foto_bukti');
259.          // dd($file);
260.          $filename = Uuid::generate(4)-
      >string.'.'.$file->getClientOriginalExtension();
261.          $path = public_path().'/img/foto_bukti';
262.          $file->move($path,$filename);
263.
264.          $this->transaksi-
      >kirimBukti($id_transaksi, $filename);
265.
266.          return redirect()->back();
267.
268.      }
269.
270.      public function detailTransaksi($id_transaksi){

271.          $detail =[
272.              'transaksi'=>DB::select('call sp_get_ba
      rang_by_transaksi(?)', [$id_transaksi])
273.          ];
274.          // dd($detail);
275.          return view('pembeli.components.detail-
      transaksi')
276.              ->with(compact('detail'))
277.              ->render();
```

```
278.        }
279.
280.        public function konfirmasiTransaksi($id_transak
        si){
281.            $this->transaksi-
        >konfirmasiTransaksi($id_transaksi);
282.            // dd($detail);
283.            return redirect()->back();
284.        }
285.    }
```

Kode Sumber 5.46 HomeController

5.3.2. Role Penjual

merupakan kumpulan *Controller* yang dikhususkan untuk role Penjual. Berikut adalah beberapa *Controller*:

5.3.2.1. BayarPetaniController

Lapisan ini bertugas untuk melakukan pembayaran ke Petani. Kode Sumber 5.47 berikut merupakan implementasi dari lapisan kontrol BayarPetaniController:

```
1.    <?php
2.
3.    namespace App\Http\Controllers\Penjual;
4.
5.    use App\Http\Controllers\Controller;
6.    use Illuminate\Http\Request;
7.    use App\Repositories\TransaksiRepository;
8.
9.    use Illuminate\Support\Facades\Storage;
10.   use Response;
11.   use DB;
12.
13.
```

```
14.  class BayarPetaniController extends Controller
15.  {
16.
17.      protected $transaksiRepository;
18.
19.      public function __construct(TransaksiRepository $
     transaksiRepository)
20.      {
21.          $this-
     >transaksiRepository = $transaksiRepository;
22.
23.          $this->middleware('auth');
24.          $this->middleware('penjual');
25.
26.      }
27.
28.      public function showAll($date){
29.          $transaksi=DB::select('call sp_get_total_baya
     r_veggo_by_tanggal(?)',array($date));
30.          $total=DB::select('call sp_get_total_bayar_ve
     ggo_by_petani(?)',array($date));
31.          $tanggal=$this->transaksiRepository-
     >getTglPetani();
32.          // dd($transaksi);
33.
34.          $data=[
35.              'filter_tanggal' =>$date,
36.              'transaksi' => $transaksi,
37.              'tanggal' => $tanggal,
38.              'total' => $total
39.          ];
40.
41.          return view('penjual.pembayaran')-
     >with('data',$data);
42.      }
43.
44.  }
```

Kode Sumber 5.47 BayarPetaniController

5.3.2.2. EtalaseController

Lapisan ini bertugas untuk mengatur etalase Penjual. Kode Sumber 5.48 berikut merupakan implementasi dari lapisan kontrol EtalaseController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Penjual;
4.
5.  use App\Http\Controllers\Controller;
6.  use Illuminate\Http\Request;
7.  use App\Repositories\ProdukRepository;
8.  use App\Repositories\KategoriRepository;
9.  use App\Repositories\BaseKategoriRepository;
10. use App\Repositories\EtalaseRepository;
11.
12. use DB;
13.
14. class EtalaseController extends Controller
15. {
16.     //
17.
18.     protected $produkRepository, $kategoriRepositor
    y, $baseKategoriRepository, $etalaseRepository;
19.
20.     public function __construct(ProdukRepository $p
    rodukRepository, KategoriRepository $kategoriReposi
    tory, BaseKategoriRepository $baseKategoriRepositor
    y, EtalaseRepository $etalaseRepository)
21.     {
22.         $this-
    >produkRepository = $produkRepository;
23.         $this-
    >kategoriRepository = $kategoriRepository;
```

```
24.             $this->baseKategoriRepository = $baseKategoriRepository;
25.             $this->etalaseRepository = $etalaseRepository;
26.
27.             $this->middleware('auth');
28.             $this->middleware('penjual');
29.
30.         }
31.
32.         public function showEtalase()
33.         {
34.             $data = [
35.                 'barang'        => $this->produkRepository->all(),
36.                 'baseKategori'  => $this->baseKategoriRepository->all(),
37.                 'kategori'      => $this->kategoriRepository->all()
38.             ];
39.             // dd($collecion);
40.
41.             if($this->keranjang->findKeranjang() == null)
42.             {
43.                 $this->keranjang->tambahKeranjang();
44.             }
45.
46.             return view('pembeli.etalase')->with(compact('data'));
47.         }
48.
49.         public function etalase(){
50.
51.             $kategoris = $this->baseKategoriRepository->all();
```

```
52.
53.             // dd($kategoris);
54.
55.             $data = [];
56.
57.             foreach($kategoris as $key => $kategori){
58.                 $data[$key]['kategori'] = $kategori-
    >kategori;
59.                 // $data[$key]['data'] = $this-
    >produkRepository-
    >getEtalaseBarangByIdKategori($kategori->id);
60.                 $data[$key]['data'] = $this-
    >etalaseRepository->findByKategori($kategori-
    >id);
61.
62.             }
63.
64.             $collection = collect($data);
65.             // dd($collection);
66.
67.             return view('penjual.etalase')-
    >with('collection', $collection);
68.         }
69.
70.         public function kelolaEtalase(){
71.             if(isset($_GET['kategori'])){
72.                 $id_kategori = $_GET['kategori'];
73.                 session(['search_kategori' => $id_kateg
    ori]);
74.             }
75.             else if(isset($_GET['nama'])){
76.                 $nama_barang = $_GET['nama'];
77.                 session(['search_nama_barang' => $nama_
    barang]);
78.             }
79.
80.
```

```
81.             $getAllBarang = $this->produkRepository-
    >findByIdKategoriOrNama(session('search_kategori'),
    session('search_nama_barang'));
82.             $getKategori = $this-
    >baseKategoriRepository->all();
83.
84.             $data = [
85.                 "kategoris" > $getKategori,
86.                 "barang" => $getAllBarang,
87.                 "id_kategori" => session('search_katego
    ri'),
88.                 "nama_barang" => session('search_nama_b
    arang')
89.             ];
90.
91.             return view('penjual.tambah_etalase')-
    >with('data',$data)-
    >with('kategoris',$getKategori);
92.
93.         }
94.
95.         public function _kelolaEtalase(Request $request
    ){
96.
97.             $updateBarangs = $request->etalase;
98.
99.             if($updateBarangs != null){
100.                foreach($updateBarangs as $updateBarang
    ){
101.                    $this->etalaseRepository-
    >updateShowEtalase($updateBarang,1);
102.                }
103.            }
104.
105.            return redirect()->back();
106.
107.
```

```
108.        }
109.
110.    }
```

Kode Sumber 5.48 EtalaseController

5.3.2.3. KategoriController

Lapisan ini bertugas untuk mengatur kategori Penjual. Kode Sumber 5.49 berikut merupakan implementasi dari lapisan kontrol KategoriController:

```
1.      <?php
2.
3.      namespace App\Http\Controllers\Penjual;
4.
5.      use App\Http\Controllers\Controller;
6.      use Illuminate\Http\Request;
7.      use App\Repositories\KategoriRepository;
8.      use App\Repositories\BaseKategoriRepository;
9.      use App\Repositories\ProdukGroupRepository;
10.     use Response;
11.
12.
13.     class KategoriController extends Controller
14.     {
15.
16.         protected $kategoriRepository, $baseKategoriRep
        ository, $produkGroupRepository;
17.
18.         public function __construct(KategoriRepository
        $kategoriRepository, BaseKategoriRepository $baseKa
        tegoriRepository,ProdukGroupRepository $produkGroup
        Repository)
19.         {
20.             $this-
        >kategoriRepository = $kategoriRepository;
```

```
21.             $this->baseKategoriRepository = $baseKategoriRepository;
22.             $this->produkGroupRepository = $produkGroupRepository;
23.
24.             $this->middleware('auth');
25.             $this->middleware('penjual');
26.
27.         }
28.
29.         public function listKategori(){
30.
31.             $getAllKategori = $this->kategoriRepository->all();
32.             $getBaseKategori = $this->baseKategoriRepository->all();
33.
34.             $data = [
35.                 "kategoris" => $getAllKategori,
36.                 "baseKategoris" => $getBaseKategori,
37.             ];
38.             return view('penjual.kategori')->with('data',$data);
39.         }
40.
41.         public function _tambahKategori(Request $request){
42.
43.             $this->kategoriRepository->create($request->all());
44.             return redirect()->back();
45.         }
46.
47.         public function _hapusKategori(Request $request){
48.             $id = $request->id;
```

```
49.             $this->kategoriRepository->delete($id);
50.             return redirect()->back();
51.         }
52.
53.         public function _ubahKategori(Request $request)
    {
54.             $id = $request->id;
55.             $this->kategoriRepository-
    >update($id, $request->all());
56.             return redirect()->back();
57.         }
58.
59.
60.     }
```

Kode Sumber 5.49 KategoriController

5.3.2.4. KlaimController

Lapisan ini bertugas untuk mengatur klaim Penjual. Kode Sumber 5.50 berikut merupakan implementasi dari lapisan kontrol KlaimController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Penjual;
4.
5.  use App\Http\Controllers\Controller;
6.  use App\Repositories\TransaksiRepository;
7.  use App\Repositories\ProdukRepository;
8.  use App\Repositories\KlaimRepository;
9.  use App\Repositories\DetailKlaimRepository;
10. use Illuminate\Http\Request;
11. use \stdClass;
12. use DB;
13. use Auth;
14. use Uuid;
```

```
15.
16.     class KlaimController extends Controller
17.     {
18.         //
19.         protected $transaksiRepository,
20.                   $klaimRepository,
21.                   $detailKlaimRepository,
22.                   $produkRepository;
23.
24.
25.         public function __construct(TransaksiRepository
    $transaksiRepository, ProdukRepository $produkRepos
    itory,KlaimRepository $klaimRepository,DetailKlaimR
    epository $detailKlaimRepository)
26.         {
27.             $this-
    >transaksiRepository = $transaksiRepository;
28.             $this-
    >produkRepository = $produkRepository;
29.             $this-
    >klaimRepository = $klaimRepository;
30.             $this-
    >detailKlaimRepository = $detailKlaimRepository;
31.             $this->middleware('auth');
32.         }
33.
34.         public function klaim(){
35.             $data = [
36.                 'klaim' => $this->klaimRepository-
    >all()
37.             ];
38.
39.             // dd($data['klaim']->dari);
40.
41.             return view('penjual.klaim')-
    >with('data',$data);
42.
```

```
43.         }
44.
45.         public function detailKlaim($id){
46.             $data = [
47.                 'klaim' => $this->detailKlaimRepository->getDetailKlaimById($id),
48.                 'klaims' => $this->klaimRepository->getByIdTrx($id)
49.             ];
50.             // dd($data['klaims']);
51.
52.             return view('penjual.ubah_klaim')->with('data', $data);
53.         }
54.
55.         public function tambahKlaim(){
56.
57.             $nomor_invoice = $this->transaksiRepository->getTransaksiSudahKirim();
58.             // dd($nomor_invoice[0]->nomor_invoice);
59.
60.             if(session('filter_tambah_klaim_invoice')){
61.                 $id = session('filter_tambah_klaim_invoice');
62.             }else{
63.                 $id = $nomor_invoice[0]->nomor_invoice;
64.             }
65.
66.             $transaksi = $this->transaksiRepository->findByNomorInvoice($id);
67.             $detailTransaksi = $transaksi->detailTransaksi;
68.
69.             foreach ($detailTransaksi as $key => $value) {
```

```
70.                 $detailTransaksi[$key]->nama_barang = $value->barang->nama;
71.             }
72.
73.             $data = [
74.                 'list_invoice' => $nomor_invoice,
75.                 'transaksi' => $transaksi,
76.                 'detailTransaksi_json' => json_encode($detailTransaksi),
77.                 'detailTransaksi' => $detailTransaksi
78.             ];
79.
80.             // dd($data);
81.             return view('penjual.tambah_klaim')->with('data',$data);
82.         }
83.
84.         public function _filterTambahKlaim(Request $request){
85.
86.             session(['filter_tambah_klaim_invoice' => $request->input('nomor_invoice')]);
87.             return redirect('/Penjual/Klaim/Tambah');
88.
89.         }
90.
91.         public function _tambahKlaim(){
92.
93.         }
94.
95.         public function _ubahKlaim(){
96.
97.         }
98.
99.         public function klaimOrderPetani($id){
100.             $transaksi = $this->transaksiRepository->findByNomorInvoice($id);
```

```
101.          $detailTransaksi = $transaksi-
      >detailTransaksi;
102.
103.          foreach ($detailTransaksi as $key => $value
      ) {
104.              $detailTransaksi[$key]-
      >nama_barang = $value->barang->nama;
105.          }
106.
107.          $data = [
108.              'transaksi' => $transaksi,
109.              'detailTransaksi_json' => json_encode($
      detailTransaksi),
110.              'detailTransaksi' => $detailTransaksi
111.          ];
112.
113.          // dd($data['detailTransaksi']);
114.          return view('penjual.tambah_klaim_orderpeta
      ni')->with('data',$data);
115.
116.      }
117.
118.      public function _klaimOrderPetani(Request $requ
      est){
119.          // dd($request->input('nama'));
120.          $klaim = new stdClass();
121.          $arr = array();
122.          for($i=0;$i<count($request-
      >input('nama'));$i++){
123.              $klaim->nama = $request-
      >input('nama')[$i];
124.              $klaim->volume = $request-
      >input('volume')[$i];
125.              $klaim->keterangan = $request-
      >input('keterangan')[$i];
126.              $klaim->foto_bukti = $request-
      >file('bukti')[$i];
```

```
127.                $klaim->id_detail_transaksi=$request->input('id_detail_transaksi')[$i];
128.
129.                // $arr[$i]=$klaim;
130.                // dd($arr[0]);
131.                array_push($arr,$klaim);
132.                $klaim = clone $klaim;
133.            }
134.            // dd($arr);
135.
136.            $data = [
137.                'id_transaksi' => $request->input('id_transaksi'),
138.                'klaim_from' => Auth::user()->id,
139.                'klaim_to' => $request->input('id_petani'),
140.                'tanggal_kirim' => $request->input('tanggal_kirim'),
141.                'status' => 1,
142.            ];
143.
144.
145.            try {
146.                //code...
147.                DB::begintransaction();
148.                $result = $this->klaimRepository->create($data);
149.                foreach($arr as $detail)
150.                {
151.                    $filename = Uuid::generate(4)->string.'.'.$detail->foto_bukti->getClientOriginalExtension();
152.                    $path = public_path().'/img/bukti_klaim';
153.                    $detail->foto_bukti->move($path,$filename);
154.                    $detailKlaim = [
```

```
                        'id_klaim' => $result->id,
                        'id_barang' => $detail->nama,
                        'id_detail_transaksi' =>$detail
    ->id_detail_transaksi,
                        'volume_klaim' => $detail-
    >volume,
                        'keterangan' => $detail-
    >keterangan,
                        'foto_bukti' => $filename,
                    ];
                    $this->detailKlaimRepository-
    >create($detailKlaim);
                }
                DB::commit();

                return redirect('/Penjual/OrderPetani/K
    onfirmasi/Terima/'.$request->nomor_invoice);
            } catch (\Throwable $th) {
                //throw $th;
                dd($th);
                DB::rollback();
            }



        }

}
```

Kode Sumber 5.50 KlaimController

5.3.2.5. OrderPetaniController

Lapisan ini bertugas untuk mengatur order Petani dari Penjual. Kode Sumber 5.51 berikut merupakan implementasi dari lapisan kontrol OrderPetaniController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Penjual;
4.
5.  use App\Http\Controllers\Controller;
6.  use App\Repositories\TransaksiRepository;
7.  use App\Repositories\UserRepository;
8.  use App\Repositories\DetailTransaksiRepository;
9.  use App\Repositories\ProdukRepository;
10. use App\Repositories\InventarisRepository;
11. use App\Repositories\HariPengirimanRepository;
12. use App\Repositories\DetailKlaimRepository;
13. use Illuminate\Http\Request;
14. use Carbon\Carbon;
15. use App\Repositories\TanggalRepository;
16. use DB;
17.
18. class OrderPetaniController extends Controller
19. {
20.     private $transaksiRepository,
21.             $detailTransaksiRepository,
22.             $produkRepository,
23.             $userRepository,
24.             $hariPengirimanRepository,
25.             $inventarisRepository,
26.             $detailKlaimRepository,
27.             $tanggal;
28.
29.     public function __construct(TransaksiRepository
    $transaksiRepository, DetailTransaksiRepository $de
    tailTransaksiRepository,InventarisRepository $inven
    tarisRepository, ProdukRepository $produkRepository
    , UserRepository $userRepository,HariPengirimanRepo
    sitory $hariPengirimanRepository,DetailKlaimReposit
    ory $detailKlaimRepository, TanggalRepository $tang
    gal)
30.     {
```

```
31.             $this->transaksiRepository = $transaksiRepository;
32.             $this->detailTransaksiRepository = $detailTransaksiRepository;
33.             $this->inventarisRepository = $inventarisRepository;
34.             $this->produkRepository = $produkRepository;
35.             $this->userRepository = $userRepository;
36.             $this->hariPengirimanRepository = $hariPengirimanRepository;
37.             $this->detailKlaimRepository = $detailKlaimRepository;
38.             $this->tanggal     = $tanggal;
39.             $this->middleware('auth');
40.         }
41.
42.         public function orderPetani()
43.         {
44.             $data = [
45.                 'orderPetani' => $this->transaksiRepository->findByTipeTransaksi("FROM_VEGGO")
46.             ];
47.
48.             return view('penjual.orderpetani')->with('data',$data);
49.         }
50.
51.         public function konfirmasiPenerimaan($invoice)
52.         {
53.             $data = [
54.                 'invoice' => $invoice,
```

```
55.                 'orderPetani' => $this->transaksiRepository->findByNomorInvoice($invoice)
56.             ];
57.
58.             foreach ($data['orderPetani']->detailTransaksi as $key => $value) {
59.                 $det=$this->detailKlaimRepository->getDetailKlaimByIdDetailTransaksi($data['orderPetani']->detailTransaksi[$key]['id']);
60.                 $count=0;
61.                 foreach ($det as $dett) {
62.                     $count+=$dett->volume_klaim;
63.                 }
64.                 // dd($data['orderPetani']->detailTransaksi[$key]['id']);
65.
66.                 $data['orderPetani']->detailTransaksi[$key]['klaim']=$count;
67.             }
68.             // dd($data);
69.             return view('penjual.konfirmasi_orderpetani')->with('data',$data);
70.         }
71.
72.         public function _konfirmasiPenerimaan(Request $request)
73.         {
74.             // dd($request->input());
75.             $detailTransaksi = $request->input('id_detail_transaksi');
76.             $barang = $request->input('id_barang');
77.             $volumeTerima = $request->input('volume_terima');
78.             $bobotTerima=$request->input('bobot_terima');
```

```
79.             $id_transaksi = $request-
    >input('id_transaksi');
80.             $selisih_terima=$request-
    >input('selisih_terima');
81.             try {
82.                 DB::beginTransaction();
83.                 $flag = 3;
84.                 $this->transaksiRepository-
    >updateStatusOrderKePetani($request-
    >input('id_transaksi'),$flag);
85.                 $this->transaksiRepository-
    >updateTanggalTerima($request-
    >input('id_transaksi'),Carbon::now());
86.                 foreach ($detailTransaksi as $key => $v
    alue) {
87.                     $id = $detailTransaksi[$key];
88.                     $id_barang = $barang[$key];
89.                     $value = $volumeTerima[$key];
90.                     $value2=$bobotTerima[$key];
91.                     $value3=$selisih_terima[$key];
92.                     $this->detailTransaksiRepository-
    >updatePenerimaanOrderKePetani($id,$value, $value2,
    $value3, $flag);
93.                     $this->inventarisRepository-
    >inventarisIn($id_barang,$id_transaksi,$value*$valu
    e2,"Order Ke Petani");
94.                     $this->produkRepository-
    >addStok($id_barang,$value*$value2);
95.                 }
96.                 DB::commit();
97.             } catch (\Throwable $th) {
98.                 DB::rollback();
99.                 dd($th);
100.            }
101.
102.            return redirect()->back();
103.        }
```

```
104.
105.       private function getTanggalPengiriman()
106.       {
107.           $getListHari=$this->tanggal-
      >get_tanggal();
108.           // dd(count($getListHari));
109.
110.           // $finalListHari=null;
111.           // dd($getListHari);
112.           if(count($getListHari)>0){
113.           for($a=0;$a<sizeof($getListHari);$a++)
114.           {
115.               $tgl=$getListHari[$a]['tanggal'];
116.               // string
117.               $finalListHari[$a]['tanggal'] = Carbon:
      :parse($tgl)->format('D, d F Y');
118.               //tanggal value
119.               $finalListHari[$a]['tanggal_value'] = C
      arbon::parse($tgl)->format('Y-m-d');
120.
121.               // dd($finalListHari[$a]['tanggal_value
      ']);
122.           }
123.           }
124.           else{
125.               $finalListHari[0]['tanggal'] = Carbon::
      now()->format('D, d F Y');
126.               //tanggal value
127.               $finalListHari[0]['tanggal_value'] = Ca
      rbon::now()->format('Y-m-d');
128.           }
129.
130.           // dd($finalListHari);
131.           foreach ($finalListHari as $key => $value)

132.           {
```

```
133.                 $finalListHari[$key] = (object) $value;

134.             }
135.
136.             // dd($finalListHari);
137.
138.             return $finalListHari;
139.         }
140.
141.         public function tambahOrderPetani(){
142.
143.             $data = [
144.                 'barang' => $this->produkRepository-
    >getBarangTanpaPenjual(),
145.                 'tanggal_kirim' => $this-
    >getTanggalPengiriman(),
146.             ];
147.
148.             // dd($data);
149.
150.             return view('penjual.tambah_orderpetani')-
    >with('data',$data);
151.         }
152.
153.         public function _tambahOrderPetani(Request $req
    uest){
154.             $volume_arr = $request->input('volume');
155.             $bobot_arr = $request->input('bobot');
156.             $barang_arr = $request->input('barang');
157.             $tanggal = $request-
    >input('tanggal_kirim');
158.             $barang = [];
159.             $supplier = [];
160.
161.             // dd($tanggal);
162.
163.             for($i=0;$i<count($barang_arr);$i++){
```

```
164.                $data = $this->produkRepository->findById($barang_arr[$i]);
165.                $data->volume = $volume_arr[$i];
166.                $data->bobot = $bobot_arr[$i];
167.                array_push($barang,$data);
168.            }
169.            // dd($barang);
170.
171.            foreach($barang as $item){
172.                if(in_array($item->id_user,$supplier)){
173.
174.                }else{
175.                    array_push($supplier,$item->id_user);
176.                }
177.            }
178.
179.            foreach($supplier as $s_key => $s){
180.                $petani = $this->userRepository->find($s);
181.                $kePetani = [
182.                    'id_user'=> $s,
183.                    'petani' => $petani,
184.                    'tanggal' => $tanggal
185.                ];
186.                $transaksi = $this->transaksiRepository->orderKePetani($kePetani);
187.                foreach($barang as $b_key => $d){
188.                    if($d->id_user == $supplier[$s_key]){
189.                        $detailKePetani = [
190.                            'id_transaksi' => $transaksi->id,
191.                            'barang' => $d,
192.                            'akumulasi_barang' => $d
193.                        ];
```

```
194.                     $this->detailTransaksiRepository->detailOrderKePetani($detailKePetani);
195.                 }
196.             }
197.         }
198.         // dd([$arr,$supplier]);
199.         return redirect()->back();
200.         // $this->transaksiRepository->orderKePetani();
201.     }
202.
203. }
204. ;
```

Kode Sumber 5.51 OrderPetaniController

5.3.2.6. PaketController

Lapisan ini bertugas untuk mengatur Paket Penjual. Kode Sumber 5.52 berikut merupakan implementasi dari lapisan kontrol PaketController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Penjual;
4.
5.  use App\Http\Controllers\Controller;
6.  use App\Repositories\ProdukRepository;
7.  use App\Repositories\KategoriRepository;
8.  use App\Repositories\BaseKategoriRepository;
9.  use App\Repositories\FotoProdukRepository;
10. use App\Repositories\IsiPaketRepository;
11. use App\Repositories\ProdukGroupRepository;
12. use App\Repositories\UserRepository;
13. use Illuminate\Http\Request;
14. use Uuid;
```

```
15.  use Auth;
16.  use View;
17.  use DB;
18.
19.
20.  class PaketController extends Controller
21.  {
22.      //
23.      protected $produkRepository, $kategoriRepository, $baseKategoriRepository, $fotoProdukRepository, $isiPaketRepository, $produkGroupRepository;
24.
25.      public function __construct(ProdukRepository $produkRepository, KategoriRepository $kategoriRepository, BaseKategoriRepository $baseKategoriRepository, FotoProdukRepository $fotoProdukRepository, IsiPaketRepository $isiPaketRepository, ProdukGroupRepository $produkGroupRepository, UserRepository $userRepository)
26.      {
27.          $this->produkRepository = $produkRepository;
28.          $this->kategoriRepository = $kategoriRepository;
29.          $this->baseKategoriRepository = $baseKategoriRepository;

30.          $this->fotoProdukRepository = $fotoProdukRepository;
31.          $this->isiPaketRepository = $isiPaketRepository;
32.          $this->produkGroupRepository = $produkGroupRepository;
33.          $this->userRepository = $userRepository;
34.
35.          $this->middleware('auth');
36.
```

```
37.         }
38.         public function listPaket()
39.         {
40.             $getAllProduk = $this->produkRepository-
    >getPaket();
41.
42.             return view('penjual.paket')-
    >with('produks', $getAllProduk);
43.         }
44.
45.         private function uploadPhoto($id,$datas){
46.
47.             $fotos = $this->fotoProdukRepository-
    >findByIdBarang($id);
48.
49.             foreach($fotos as $foto){
50.                 $path = public_path().'\img\foto_barang
    \\';
51.                 $filename = $path.$foto->path;
52.                 if(file_exists($filename)){
53.                     unlink($filename);
54.                     $this->fotoProdukRepository-
    >delete($id,$foto->path);
55.                 }
56.             }
57.
58.
59.             foreach($datas as $data){
60.                 $filename = time().$data-
    >getClientOriginalName();
61.                 $path = public_path().'/img/foto_barang
    ';
62.                 $data->move($path,$filename);
63.                 $this->fotoProdukRepository-
    >create($id, $filename);
64.             }
65.
```

```
66.         }
67.
68.
69.         private function generateKode(){
70.
71.             $finalStr = 'BP'.rand(100,999);
72.
73.             $kode = $this->produkRepository-
    >findByKode($finalStr);
74.             if($kode){
75.                 $this->generateKode();
76.             }else{
77.                 return $finalStr;
78.             }
79.         }
80.
81.         public function tambahPaket(){
82.             $produk = $this->produkRepository->all();
83.             $subKategori = $this->kategoriRepository-
    >all();
84.             $kategori = $this->baseKategoriRepository-
    >all();
85.             $getAllSupplier = $this->userRepository-
    >getPetani();
86.
87.             $data = [
88.                 'produk' => $produk,
89.                 'subKategori' => $subKategori,
90.                 'kategori' => $kategori,
91.                 'supplier' => $getAllSupplier
92.             ];
93.
94.             return view('penjual.tambah_paket')-
    >with('data',$data);
95.         }
96.
```

```
97.         public function _tambahPaket(Request $request){

98.             // dd($request->file('foto_barang'));
99.             if($request->jenis_diskon=="Potongan Persen"){
100.                 if ($request->jenis_diskon_reseller=="Potongan Persen") {
101.                     $harga_diskon=$request->harga_jual-($request->harga_jual*($request->diskon/100));
102.                     $reseller=($harga_diskon-($harga_diskon*($request->diskon_reseller/100)));
103.                 } else {
104.                     $harga_diskon=$request->harga_jual-($request->harga_jual*($request->diskon/100));
105.                     $reseller=$harga_diskon-$request->diskon_reseller;
106.                 }
107.             }
108.             else{
109.                 if ($request->jenis_diskon_reseller=="Potongan Persen") {
110.                     $harga_diskon=$request->harga_jual-$request->diskon;
111.                     $reseller=($harga_diskon-($harga_diskon*($request->diskon_reseller/100)));
112.                 } else {
113.                     $harga_diskon=$request->harga_jual-$request->diskon;
114.                     $reseller=$harga_diskon-$request->diskon_reseller;
115.                 }
116.             }
117.
118.             try {
119.
120.                 DB::beginTransaction();
121.                 if($request->diskon==null){
```

```
122.                     $diskon=0;
123.                 }
124.                 else{
125.                     $diskon=$request->diskon;
126.                 }
127.
128.                 if(count($request->file('foto_barang')) > 5){
129.                     return redirect()->back()->with('error',"Maksimal Upload 5 Foto!")->withInput();
130.                 }
131.
132.                 $id_barang = Uuid::generate(4)->string;
133.                 // dd($request->harga_beli);
134.                 $newProduk = [
135.                     'id' => $id_barang,
136.                     'id_user' => $request->id_user,
137.                     'id_kategori' => $request->id_kategori,
138.                     'nama' => $request->nama,
139.                     'kode' => $this->generateKode(),
140.                     'jenis' => "Paket",
141.                     'satuan' => "Pcs",
142.                     'bobot' => 1,
143.                     'harga_beli' => $request->harga_beli,
144.                     'harga_jual' => $request->harga_jual,
145.                     'deskripsi' => "-",
146.                     'diskon' => $diskon,
147.                     'jenis_diskon' => $request->jenis_diskon,
148.                     'jenis_diskon' => $request->jenis_diskon_reseller,
```

```
149.                     'diskon' => $request->diskon_reseller,
150.                     'show_etalase' => 0,
151.                     'is_paket' => 1,
152.                     'ketersediaan' => 0,
153.                     'stok'  => 0,
154.                     'harga_jual_reseller' => $reseller,
155.                 ];
156.
157.                 $item = [
158.                     'isi_paket' => $request->isi_paket,
159.                     'harga_isi_paket' => $request->harga_isi_paket,
160.                     'volume_isi_paket' => $request->volume_isi_paket,
161.                 ];
162.
163.                 $this->produkRepository->create_paket($newProduk);
164.                 $this->isiPaketRepository->create($id_barang,$item);
165.                 $this->produkGroupRepository->create($id_barang, $request->group);
166.                 $this->uploadPhoto($id_barang,$request->file('foto_barang'));
167.
168.                 DB::commit();
169.                 return redirect()->back()->with('success',"Berhasil Menambah!");
170.
171.             } catch (\Throwable $th) {
172.                 //throw $th;
173.                 DB::rollback();
174.                 dd($th);
```

```
175.          }
176.
177.      }
178.
179.      public function ubahPaket($id){
180.
181.          $paket = $this->produkRepository-
      >findById($id);
182.
183.          $produk = $this->produkRepository->all();
184.          $subKategori = $this->kategoriRepository-
      >all();
185.          $kategori = $this->baseKategoriRepository-
      >all();
186.
187.          $barangGroup = $this-
      >produkGroupRepository->findByIdBarang($id);
188.          $isiPaket = $this->isiPaketRepository-
      >findByIdBarang($id);
189.
190.          $getAllSupplier = $this->userRepository-
      >getPetani();
191.
192.          for($i=0;$i<count($subKategori);$i++){
193.              for($j=0;$j<count($barangGroup);$j++){

194.                  if(($subKategori[$i]-
      >id == $barangGroup[$j]-
      >id_kategori) && ($subKategori[$i]-
      >selected != 1)){
195.                      $subKategori[$i]-
      >selected = 1;
196.                  }elseif($subKategori[$i]-
      >selected == 0){
197.                       $subKategori[$i]-
      >selected = 0;
198.                  }
```

```
199.                 }
200.             }
201.
202.             $index = count($isiPaket);
203.
204.             $data = [
205.                 'paket' => $paket,
206.                 'isiPaket' => $isiPaket,
207.                 'produk' => $produk,
208.                 'subKategori' => $subKategori,
209.                 'kategori' => $kategori,
210.                 'index' => $index - 1,
211.                 'supplier'=>$getAllSupplier
212.             ];
213.             // dd($data);
214.
215.             return view('penjual.ubah_paket')-
    >with('data',$data);
216.
217.         }
218.
219.         public function _ubahPaket(Request $request){
220.             // dd($request->input());
221.
222.             if($request-
    >jenis_diskon=="Potongan Persen"){
223.                 if ($request-
    >jenis_diskon_reseller=="Potongan Persen") {
224.                     $harga_diskon=$request->harga_jual-
    ($request->harga_jual*($request->diskon/100));
225.                     $reseller=($harga_diskon-
    ($harga_diskon*($request->diskon_reseller/100)));
226.                 } else {
227.                     $harga_diskon=$request->harga_jual-
    ($request->harga_jual*($request->diskon/100));
228.                     $reseller=$harga_diskon-$request-
    >diskon_reseller;
```

```
229.                 }
230.             }
231.             else{
232.                 if ($request->jenis_diskon_reseller=="Potongan Persen") {
233.                     $harga_diskon=$request->harga_jual-$request->diskon;
234.                     $reseller=($harga_diskon-($harga_diskon*($request->diskon_reseller/100)));
235.                 } else {
236.                     $harga_diskon=$request->harga_jual-$request->diskon;
237.                     $reseller=$harga_diskon-$request->diskon_reseller;
238.                 }
239.             }
240.             // dd($reseller);
241.             $request->merge([
242.                 'harga_jual_reseller' => $reseller,
243.             ]);
244.             // dd($request->harga_jual_reseller);
245.
246.             // dd($request->all());
247.             DB::beginTransaction();
248.             try {
249.
250.                 $item = [
251.                     'isi_paket' => $request->isi_paket,
252.                     'harga_isi_paket' => $request->harga_isi_paket,
253.                     'volume_isi_paket' => $request->volume_isi_paket,
254.                 ];
255.
256.
```

```
257.                 $this->isiPaketRepository->create($request->id,$item);
258.                 $this->produkGroupRepository->delete($request->id);
259.                 $this->produkRepository->update($request->id,$request->all());
260.                 $this->produkGroupRepository->create($request->input('id'), $request->input('group'));
261.
262.                 DB::commit();
263.                 return redirect()->back()->with('success',"Berhasil Mengubah!")->withInput();
264.
265.             } catch (\Throwable $th) {
266.                 DB::rollback();
267.                 dd($th);
268.                 return redirect()->back()->with('error',"Gagal menambahkan barang!")->withInput();
269.
270.             }
271.
272.             return redirect()->back();
273.
274.
275.         }
276.
277.         public function getPaketItem($jumlah)
278.         {
279.             $produk = $this->produkRepository->getProduct();
280.             $data = [
281.                 'produk' => $produk,
282.                 'jumlah' => $jumlah
283.             ];
```

```
284.
285.            return View::make('penjual.layouts.item_pak
    et')
286.                ->with('data', $data)
287.                ->render();
288.        }
289.
290.
291.    }
```

Kode Sumber 5.52 PaketController

5.3.2.7. PengaturanController

Lapisan ini bertugas untuk mengatur pengaturan Penjual. Kode Sumber 5.53 berikut merupakan implementasi dari lapisan kontrol PengaturanController:

```
1.      <?php
2.
3.      namespace App\Http\Controllers\Penjual;
4.
5.      use App\Http\Controllers\Controller;
6.      use Illuminate\Http\Request;
7.      use App\Repositories\ProdukRepository;
8.      use App\Repositories\UserRepository;
9.      use App\Repositories\KategoriRepository;
10.     use App\Repositories\BaseKategoriRepository;
11.     use App\Repositories\BobotKemasanRepository;
12.     use App\Repositories\ProdukKemasanRepository;
13.     use App\Repositories\ProdukGroupRepository;
14.     use App\Repositories\FotoProdukRepository;
15.     use App\Repositories\InventarisRepository;
16.     use App\Repositories\TanggalRepository;
17.     use App\Repositories\BarangTanggalRepository;
18.
19.     use Illuminate\Support\Facades\Storage;
```

```
20.    use Illuminate\Support\Facades\Hash;
21.    use Response;
22.    use DB;
23.    use Auth;
24.    use Uuid;
25.
26.
27.    class PengaturanController extends Controller
28.    {
29.
30.        protected $produkRepository, $userRepository, $
       kategoriRepository, $bobotKemasanRepository, $baseK
       ategoriRepository, $produkKemasanRepository, $produ
       kGroupRepository, $fotoProdukRepository, $inventari
       sRepository, $tanggalRepository, $barangTanggalRepo
       sitory;
31.
32.        public function __construct(ProdukRepository $p
       rodukRepository, UserRepository $userRepository, Ka
       tegoriRepository $kategoriRepository, BobotKemasanR
       epository $bobotKemasanRepository, BaseKategoriRepo
       sitory $baseKategoriRepository, ProdukKemasanReposi
       tory $produkKemasanRepository, ProdukGroupRepositor
       y $produkGroupRepository, FotoProdukRepository $fot
       oProdukRepository, InventarisRepository $inventaris
       Repository, TanggalRepository $tanggalRepository, B
       arangTanggalRepository $barangTanggalRepository)
33.        {
34.            $this-
       >produkRepository = $produkRepository;
35.            $this->userRepository = $userRepository;
36.            $this-
       >kategoriRepository = $kategoriRepository;
37.            $this-
       >bobotKemasanRepository = $bobotKemasanRepository;
```

```
38.             $this->baseKategoriRepository = $baseKategoriRepository;
39.             $this->produkKemasanRepository = $produkKemasanRepository;
40.             $this->produkGroupRepository = $produkGroupRepository;
41.             $this->fotoProdukRepository = $fotoProdukRepository;
42.             $this->inventarisRepository = $inventarisRepository;
43.             $this->tanggalRepository = $tanggalRepository;
44.             $this->barangTanggalRepository = $barangTanggalRepository;
45.
46.             $this->middleware('auth');
47.             $this->middleware('penjual');
48.
49.         }
50.
51.         public function tanggal($tanggal){
52.             // dd($tanggal);
53.             $timestamp = getdate(strtotime($tanggal));
54.
55.             $month = $timestamp['mon'];
56.             $year = $timestamp['year'];
57.
58.
59.             // Create array containing abbreviations of days of week.
60.             $daysOfWeek = array('Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday');
```

```
61.
62.             // What is the first day of the month in qu
    estion?
63.             $firstDayOfMonth = mktime(0,0,0,$month,1,$y
    ear);
64.
65.             // How many days does this month contain?
66.             $numberDays = date('t',$firstDayOfMonth);
67.
68.             // Retrieve some information about the firs
    t day of the
69.             // month in question.
70.             $dateComponents = getdate($firstDayOfMonth)
    ;
71.
72.             // What is the name of the month in questio
    n?
73.             $monthName = $dateComponents['month'];
74.
75.             // What is the index value (0-
    6) of the first day of the
76.             // month in question.
77.             $dayOfWeek = $dateComponents['wday'];
78.
79.             // Create the table tag opener and day head
    ers
80.
81.             $datetoday = date('Y-m-d');
82.
83.
84.             $calendar = "<center><h2>$monthName $year</
    h2>";
85.
86.             $calendar.= "<a class='btn btn-xs btn-
    success' href='/Penjual/Pengaturan/Tanggal/".date('
    Y-m-d',mktime(0, 0, 0, $month-
    1, 1, $year))."'>Bulan Sebelumnya</a> ";
```

```
87.
88.             $calendar.= "<a class='btn btn-xs btn-
    success' href='/Penjual/Pengaturan/Tanggal/".date('
    Y-m-d')."'>Bulan Sekarang</a> ";
89.
90.             $calendar.= "<a class='btn btn-xs btn-
    success' href='/Penjual/Pengaturan/Tanggal/".date('
    Y-m-
    d',mktime(0, 0, 0, $month+1, 1, $year))."'>Bulan Se
    lanjutnya</a> </center><br>";
91.
92.             $calendar .= "<table class='table table-
    bordered' style='overflow-x:auto;'>";
93.
94.
95.             $calendar .= "<tr>";
96.
97.             // Create the calendar headers
98.
99.             foreach($daysOfWeek as $day) {
100.                $calendar .= "<th  class='header'>$day<
    /th>";
101.            }
102.
103.            // Create the rest of the calendar
104.
105.            // Initiate the day counter, starting with
    the 1st.
106.
107.            $currentDay = 1;
108.
109.            $calendar .= "</tr><tr>";
110.
111.            // The variable $dayOfWeek is used to
112.            // ensure that the calendar
113.            // display consists of exactly 7 columns.
114.
```

```
115.            if ($dayOfWeek > 0) {
116.                for($k=0;$k<$dayOfWeek;$k++){
117.                        $calendar .= "<td  class='empty
    '></td>";
118.
119.                }
120.            }
121.
122.
123.            $month = str_pad($month, 2, "0", STR_PAD_LE
    FT);
124.
125.            while ($currentDay <= $numberDays) {
126.
127.                // Seventh column (Saturday) reached. S
    tart a new row.
128.
129.                if ($dayOfWeek == 7) {
130.
131.                    $dayOfWeek = 0;
132.                    $calendar .= "</tr><tr>";
133.
134.                }
135.
136.                $currentDayRel = str_pad($currentDay, 2
    , "0", STR_PAD_LEFT);
137.                $date = "$year-$month-
    $currentDayRel";
138.
139.                $dayname = strtolower(date('l', strtoti
    me($date)));
140.                $eventNum = 0;
141.                $today = $date==date('Y-m-
    d')? "today" : "";
142.                // dd(date('Y-m-d'));
143.
```

```
144.                 // dd($this->tanggalRepository-
      >find_tanggal($date));
145.                 if($this->tanggalRepository-
      >find_tanggal($date)->isEmpty() ){
146.                     if(strtotime($date)<strtotime(date(
      'Y-m-d'))){
147.                         $calendar.="<td class='$today'>
      <h4> $currentDay</h4> <button  class='btn btn-
      danger btn-xs' disabled>Tutup</button>";
148.                     }
149.                     else{
150.                         $calendar.="<td class='$today'>
      <h4>$currentDay</h4> <a href='/Penjual/Pengaturan/B
      ukaTanggal/".$date."' class='btn btn-danger btn-
      xs'>Tutup</a>";
151.                     }
152.                 }
153.                 else{
154.                     $calendar.="<td class='$today'><h4>
      <a href='/Penjual/Pengaturan/BukaTanggall/".$date."
      '>$currentDay</a></h4> <a href='/Penjual/Pengaturan
      /CloseTanggal/".$date."' class='btn btn-
      success btn-xs'>Buka</a>";
155.                 }
156.
157.                 $calendar .="</td>";
158.                 // Increment counters
159.
160.                 $currentDay++;
161.                 $dayOfWeek++;
162.
163.             }
164.
165.             // Complete the row of the last week in mon
      th, if necessary
166.
167.             if ($dayOfWeek != 7) {
```

```
168.
169.                 $remainingDays = 7 - $dayOfWeek;
170.                     for($l=0;$l<$remainingDays;$l++){
171.                         $calendar .= "<td class='empty'
    ></td>";
172.
173.                 }
174.
175.             }
176.
177.             $calendar .= "</tr>";
178.
179.             $calendar .= "</table>";
180.
181.             return view('penjual.tanggal')-
    >with('data', $calendar);
182.         }
183.
184.         public function bukaTanggal($tanggal){
185.             if($this->tanggalRepository-
    >find_tanggal_not_status($tanggal)->isEmpty() ){
186.                 $this->tanggalRepository-
    >create($tanggal, 1);
187.             }
188.             else{
189.                 $this->tanggalRepository-
    >update($tanggal, 1);
190.             }
191.             return redirect('/Penjual/Pengaturan/Tangga
    l/'.date("Y-m-d"));
192.         }
193.
194.         public function submitTanggal(Request $request)
    {
195.             $this->barangTanggalRepository-
    >deleteByTanggal($request->tanggal);
```

```
196.            foreach($request->etalase as $updateBarang){
197.                $this->barangTanggalRepository->create($request->tanggal, $updateBarang);
198.            }
199.            return redirect('/Penjual/Pengaturan/Tanggal/'.date("Y-m-d"));
200.        }
201.
202.        public function closeTanggal($tanggal){
203.            $this->barangTanggalRepository->deleteByTanggal($tanggal);
204.            $this->tanggalRepository->update($tanggal, 0);
205.            return redirect()->back();
206.        }
207.
208.        public function tambahUser(){
209.            return view('penjual.tambah_user');
210.        }
211.        public function _tambahUser(Request $request){

212.            $id = Uuid::generate(4)->string;
213.            $this->userRepository->create([
214.                'id' => $id,
215.                'name' => $request['name'],
216.                'email' => $request['email'],
217.                'nomor_hp'=> $request['nomor_hp'],
218.                'password' => Hash::make($request['password']),
219.                'role' => $request['role']
220.            ]);
221.            return redirect()->back();
222.        }
223.
224.        public function bukaTanggall($tanggal){
225.            $data=[
```

```
226.                 'barang' => $this->produkRepository->all(),
227.                 'tanggal' => $tanggal
228.             ];
229.             return view('penjual.buka_tanggal')->with(compact('data'));
230.             // dd($tanggal);
231.         }
232.
233.         public function rekening(){
234.             $data=$this->userRepository->detail(Auth::user()->id);
235.             // dd($data);
236.             return view('penjual.edit_rekening')->with(compact('data'));
237.         }
238.         public function _rekening(Request $request){
239.             $this->userRepository->update(Auth::user()->id, [
240.                 'nomor_rek' => $request['nomor_rek'],
241.                 'bank' => $request['bank'],
242.                 'atas_nama'=> $request['atas_nama'],
243.             ]);
244.             return redirect()->back();
245.         }
246.     }
```

Kode Sumber 5.53 PengaturanController

5.3.2.8. PengirimanController

Lapisan ini bertugas untuk mengatur pengiriman Penjual. Kode Sumber 5.54 berikut merupakan implementasi dari lapisan kontrol PengirimanController:

```
1.  <?php
2.
```

```
3.  namespace App\Http\Controllers\Penjual;
4.
5.  use App\Http\Controllers\Controller;
6.  use Illuminate\Http\Request;
7.  use Illuminate\Support\Facades\Auth;
8.  use App\Repositories\TransaksiRepository;
9.  use App\Repositories\DetailTransaksiRepository;
10. use App\Repositories\UserRepository;
11. use App\Repositories\ProdukRepository;
12. use App\Repositories\InventarisRepository;
13.
14. use DB;
15.
16.
17.
18. class PengirimanController extends Controller
19. {
20.
21.     private $transaksiRepository,
22.             $userRepository,
23.             $inventarisRepository,
24.             $produkRepository,
25.             $detailTransaksisRepository;
26.
27.     public function __construct(TransaksiRepository
    $transaksiRepository, DetailTransaksiRepository $de
    tailTransaksisRepository,UserRepository $userReposi
    tory,InventarisRepository $inventarisRepository,Pro
    dukRepository $produkRepository)
28.     {
29.         $this-
    >transaksiRepository = $transaksiRepository;
30.         $this-
    >detailTransaksisRepository = $detailTransaksisRepo
    sitory;
31.         $this->userRepository = $userRepository;
```

```
32.             $this->inventarisRepository = $inventarisRepository;
33.             $this->produkRepository = $produkRepository;
34.             $this->middleware('auth');
35.         }
36.
37.         public function pengiriman($date){
38.
39.             $tipe = "FROM_BUYER";
40.             $tanggal = $date;
41.             $status = 5;
42.             $status2 = 6;
43.
44.             $resultCount = $this->transaksiRepository->getCountVerif2($status, $status2,$tipe);
45.             $tanggals=$this->transaksiRepository->getTransactionDateWithStatus2($status, $status2, $tipe);
46.
47.             foreach($tanggals as $tgl){
48.                 $tgl->total=0;
49.                 foreach($resultCount as $resulttcount){

50.                     if($resulttcount->tanggal_pre_order==$tgl->tanggal_pre_order){
51.                         $tgl->total+=$resulttcount->total;
52.                     }
53.                     else{
54.                         $tgl->total+=0;
55.                     }
56.                 }
57.             }
58.
59.             $data = [
```

```
60.                 'preorder' => $this->transaksiRepository->findByTanggalPreOrderAndStatusAndTipeTransakis2($tanggal,$status, $status2,$tipe),
61.                 'filter_tanggal' => $date,
62.                 'tanggal' => $tanggals,
63.             ];
64.             // dd($data);
65.             return view('penjual.pengiriman')->with('data',$data);
66.         }
67. 
68.         public function tambahPengiriman(){
69.             $data = [
70.                 'kurir' => $this->userRepository->getKurir(),
71.                 'reseller' => $this->userRepository->getReseller(),
72.                 'current_date' => session('kirim_date'),
73.                 'current_kurir' => session('kirim_kurir'),
74.                 'current_reseller' => session('kirim_reseller'),
75.                 'tanggal' => $this->transaksiRepository->getTransactionDate(),
76.                 'preorder' => $this->transaksiRepository->findByTanggalPreOrderAndStatusAndTipeTransakis(session('kirim_date'),4,"FROM_BUYER"),
77.             ];
78.             return view('penjual.tambah_pengiriman')->with('data',$data);
79.         }
80. 
81.         public function _filterTambahPengiriman(Request $request){
```

```
82.             // dd($request->input());
83.             session(['kirim_date' => $request-
    >input('tanggal')]);
84.             session(['kirim_kurir' => $request-
    >input('kurir')]);
85.             return redirect()->back();
86.
87.         }
88.
89.         public function _tambahPengiriman(Request $requ
    est){
90.             // dd($request->input());
91.             $id_kurir = $request-
    >input('current_kurir');
92.             $id_reseller = $request-
    >input('current_reseller');
93.             $pengiriman = $request-
    >input('pengiriman');
94.
95.             if($pengiriman==null){
96.                 return redirect()->back();
97.             }
98.             else{
99.             try {
100.                DB::begintransaction();
101.                for($i=0;$i<count($pengiriman);$i++){
102.                    $id = $pengiriman[$i];
103.                    $kurir = $id_kurir;
104.                    $reseller = $id_reseller;
105.                    if(strlen($kurir)<30){
106.                        if($id_reseller==null){
107.                            $flag = 7;
108.                        }
109.                        else{
110.                            $flag = 6;
111.                        }
112.                    }
```

```
                    else{
                        $flag = 5;
                    }
                    $this->transaksiRepository->updateKurirPengiriman($id,$kurir, $reseller,$flag);
                    $transaksi = $this->transaksiRepository->find($id);

                    $detail = $transaksi->detailTransaksi;
                    // dd($detail[2]->barang);
                    for($j=0;$j<count($detail);$j++){
                        $barang = $detail[$j];

                        if($barang->is_canceled_by_veggo === 0){
                            $this->kelolaInventaris($barang,$transaksi,$id);
                            $this->detailTransaksisRepository->updateByIdTransaksi($transaksi->id,['status' => 5]);
                        }
                    }
                }

                DB::commit();

                return redirect()->back();
            } catch (\Throwable $th) {
                //throw $th;
                DB::rollback();
                dd($th);
            }}
        }

```

```
141.      private function kelolaInventaris($barang,$transaksi,$id){
142.          $keterangan = "PENGIRIMAN KE PEMBELI OLEH ".$transaksi->id_kurir;
143.
144.          if($barang->barang->jenis == "Timbang" && $barang->barang->is_paket == 0){
145.              $volume = (int) $barang->volume_kirim_kurir;
146.              $this->inventarisRepository->inventarisOut($barang->id_barang,$id,$volume,$keterangan);
147.              $this->produkRepository->removeStok($barang->id_barang,$volume);
148.
149.          }else if($barang->barang->is_paket == 1){
150.              $volume = (int) $barang->volume_kirim_kurir;
151.              $isipaket = $barang->barang->isiPaket;
152.              // $this->inventarisRepository->inventarisOut($barang->id_barang,$id,$volume,$keterangan);
153.              // $this->produkRepository->removeStok($barang->id_barang,$volume);
154.              for($i=0;$i<count($isipaket);$i++){
155.                  $isi = $isipaket[$i];
156.                  $this->inventarisRepository->inventarisOut($isi->id_barang,$id,$isi->volume,$keterangan);
157.                  $this->produkRepository->removeStok($isi->id_barang,(int) $isi->volume);
158.              }
```

```
159.          }else if($barang->barang->jenis == "Kemas" && $barang->barang->is_paket == 0){
160.              $volume = (int) ($barang->volume_kirim_kurir * $barang->bobot_kemasan);
161.              $this->inventarisRepository->inventarisOut($barang->id_barang,$id,$volume,$keterangan);
162.              $this->produkRepository->removeStok($barang->id_barang,$volume);
163.
164.          }else{
165.              dd("error");
166.          }
167.      }
168.
169.      public function finalisasi(){
170.
171.      }
172.
173.      public function finalisasiPengiriman($id){
174.
175.          $data = [
176.              'produk' => $this->produkRepository->all(),
177.              'transaksi' => $this->transaksiRepository->find($id)
178.          ];
179.
180.          return view('penjual.finalisasi_pengiriman')->with('data',$data);
181.
182.      }
183.
184.      public function _finalisasiPengiriman(Request $request){
185.          // dd($request->input());
```

```
186.          $idDetailtransaksi = $request-
      >input('id_detail_transaksi');
187.          $volumeKirimKurir = $request-
      >input('volume_kirim_kurir');
188.          $bobotKirimKurir = $request-
      >input('bobot_kirim_kurir');
189.          $harga_akhir = $request-
      >input('harga_akhir');
190.          $idTransaksi = $request-
      >input('id_transaksi');
191.          $keterangan = $request-
      >input('keterangan');
192.          $harga_akhir_diskon=$request-
      >input('harga_akhir_diskon');
193.          $total_harga_akhir=$request-
      >input('total_harga');
194.          $ongkir=$request->input('ongkir');
195.
196.          // $transaksi = $this->transaksiRepository-
      >find($idTransaksi);
197.
198.          // $detail = $transaksi->detailTransaksi;
199.          // dd($detail);
200.
201.          try {
202.              DB::begintransaction();
203.              $this->transaksiRepository-
      >update($idTransaksi, ['keterangan' => $keterangan,
      'total_bayar_akhir' => $total_harga_akhir, 'ongkir'
      =>$ongkir]);
204.              $this->transaksiRepository-
      >updateStatusOrderKePetani($idTransaksi,4);
205.              for($i=0;$i<count($idDetailtransaksi);$
      i++){
206.                  $id = $idDetailtransaksi[$i];
207.                  $volume = $volumeKirimKurir[$i];
208.                  $bobot = $bobotKirimKurir[$i];
```

```
209.                     $harga = $harga_akhir[$i];
210.                     $harga_diskon=$harga_akhir_diskon[$
    i];
211.                     $this->detailTransaksisRepository-
    >updateFinalisasiPengiriman($id,$volume, $bobot,$ha
    rga, $harga_diskon);
212.                 }
213.
214.                 //dari siap kirim
215.                 $this->transaksiRepository-
    >updateKurirPengiriman($idTransaksi,"id_kurir", nul
    l,7);
216.                 $transaksi = $this-
    >transaksiRepository->find($idTransaksi);
217.
218.                 $detail = $transaksi-
    >detailTransaksi;
219.                 // dd($detail[2]->barang);
220.                 for($j=0;$j<count($detail);$j++){
221.                     $barang = $detail[$j];
222.                     // dd($barang-
    >is_canceled_by_veggo);
223.
224.                     if($barang-
    >is_canceled_by_veggo == 0){
225.                         $this-
    >kelolaInventaris($barang,$transaksi,$id);
226.                         $this-
    >detailTransaksisRepository-
    >updateByIdTransaksi($transaksi-
    >id,['status' => 5]);
227.                     }
228.                 }
229.
230.                 DB::commit();
231.             } catch (\Throwable $th) {
232.                 DB::rollback();
```

```
233.                 dd($th);
234.             }
235.
236.             return redirect('Penjual/PreOrder/Selesai/T
    anggal/'.date("Y-m-d"));
237.         }
238.
239.         public function _tambahItemFinalisasiPengiriman
    (Request $request){
240.             // dd($request->input());
241.             $id_transaksi=$request-
    >input('id_transaksi');
242.             $id_barang=$request->input('id_barang');
243.             $jumlah_kirim=$request-
    >input('volume_kirim_kurir');
244.             $harga=$request->input('harga_akhir');
245.
246.             $this->detailTransaksisRepository-
    >addFinalisasiItem($id_transaksi,$id_barang,$jumlah
    _kirim,$harga);
247.             $url = "/Penjual/Pengiriman/Finalisasi/".$i
    d_transaksi;
248.             return redirect($url);
249.         }
250.
251.         public function _hapusItemFinalisasiPengiriman(
    Request $request){
252.             $this->detailTransaksisRepository-
    >removeFinalisasiItem($request-
    >input('id_detail_transaksi'));
253.         }
254.
255.
256.
257.     }
```

Kode Sumber 5.54 PengirimanController

5.3.2.9. PreOrderController

Lapisan ini bertugas untuk mengatur *Preorder* Penjual. Kode Sumber 5.55 berikut merupakan implementasi dari lapisan kontrol PreOrderController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Penjual;
4.
5.  use Illuminate\Http\Request;
6.  use App\Http\Controllers\Controller;
7.  use App\Repositories\DetailTransaksiRepository;
8.  use App\Repositories\TransaksiRepository;
9.  use App\Repositories\ProdukRepository;
10. use App\Repositories\UserRepository;
11. use App\Repositories\ParentKeranjangResellerReposit
    ory;
12. use App\Repositories\KeranjangResellerRepository;
13. use Carbon\Carbon;
14. use Uuid;
15. use Auth;
16. use View;
17. use DB;
18.
19.
20. class PreOrderController extends Controller
21. {
22.     //
23.
24.     protected $transaksiRepository,
25.               $detailTransaksiRepository,
26.               $userRepository,
27.               $produkRepository,
28.               $parentKeranjangResellerRepository,
29.               $keranjangResellerRepository;
```

```
30.
31.         public function __construct(TransaksiRepository
    $transaksiRepository, DetailTransaksiRepository $de
    tailTransaksiRepository, ProdukRepository $produkRe
    pository, UserRepository $userRepository, ParentKer
    anjangResellerRepository $parentKeranjangResellerRe
    pository, KeranjangResellerRepository $keranjangRes
    ellerRepository)
32.         {
33.             $this-
    >transaksiRepository = $transaksiRepository;
34.             $this-
    >detailTransaksiRepository = $detailTransaksiReposi
    tory;
35.             $this->userRepository = $userRepository;
36.             $this-
    >produkRepository = $produkRepository;
37.             $this-
    >parentKeranjangResellerRepository = $parentKeranja
    ngResellerRepository;
38.             $this-
    >keranjangResellerRepository = $keranjangResellerRe
    pository;
39.
40.             $this->middleware('auth');
41.             $this->middleware('penjual');
42.         }
43.
44.         public function preOrder($date){
45.             // dd("test");
46.             $tipe = "FROM_BUYER";
47.             $tanggal = $date;
48.             $status = 1;
49.             $cancel=0;
50.
```

```
51.         $result = $this->transaksiRepository->findByTanggalPreOrderAndStatusAndTipeTransakis($tanggal,$status,$tipe);
52.         $resultCount = $this->transaksiRepository->getCountVerif($status,$tipe, $cancel);
53.         $tanggals=$this->transaksiRepository->getTransactionDateWithStatus($status, $tipe, $cancel);
54.
55.         foreach($tanggals as $tgl){
56.             $tgl->total=0;
57.             foreach($resultCount as $resulttcount){

58.                 if($resulttcount->tanggal_pre_order==$tgl->tanggal_pre_order){
59.                     $tgl->total+=$resulttcount->total;
60.                 }
61.                 else{
62.                     $tgl->total+=0;
63.                 }
64.             }
65.         }
66.
67.         $data = [
68.             'preorder' => $result,
69.             'isPaid' => [["Belum Lunas","0"],["Lunas","1"]],
70.             'status' => [['Pre Order',1],['Siap Kirim',4],['Pengiriman',5],['Batal',7],['Selesai',6]],

71.             'status_pembayaran' => [['Tunggu Konfirmasi',1],['Gagal',2],['Lunas',3],['Belum Lunas',0]],
72.             'tanggal' => $tanggals,
73.             'filter_tanggal' => $tanggal
74.         ];
```

```
75.
76.
77.
78.             return view('penjual.preorder')-
    >with('data',$data);
79.         }
80.
81.         public function prosesOrder($date){
82.
83.             $tipe = "FROM_BUYER";
84.             $tanggal = $date;
85.             $status = 2;
86.             $cancel=0;
87.
88.             $result = $this->transaksiRepository-
    >findByTanggalPreOrderAndStatusAndTipeTransakis($ta
    nggal,$status,$tipe);
89.             $resultCount = $this->transaksiRepository-
    >getCountVerif($status,$tipe, $cancel);
90.             $tanggals=$this->transaksiRepository-
    >getTransactionDateWithStatus($status, $tipe, $canc
    el);
91.
92.             foreach($tanggals as $tgl){
93.                 $tgl->total=0;
94.                 foreach($resultCount as $resulttcount){

95.                     if($resulttcount-
    >tanggal_pre_order==$tgl->tanggal_pre_order){
96.                         $tgl->total+=$resulttcount-
    >total;
97.                     }
98.                     else{
99.                         $tgl->total+=0;
100.                    }
101.                }
102.            }
```

```
103.
104.           $data = [
105.               'preorder' => $result,
106.               'isPaid' => [["Belum Lunas","0"],["Luna
       s","1"]],
107.               'status' => [['Pre Order',1],['Siap Kir
       im',4],['Pengiriman',5],['Batal',7],['Selesai',6]],

108.               'status_pembayaran' => [['Tunggu Konfir
       masi',1],['Gagal',2],['Lunas',3],['Belum Lunas',0]]
       ,
109.               'tanggal' => $tanggals,
110.               'filter_tanggal' => $tanggal
111.           ];
112.
113.
114.           return view('penjual.preorder_proses')-
       >with('data',$data);
115.       }
116.
117.       public function siapKirim($date){
118.
119.           $tipe = "FROM_BUYER";
120.           $tanggal = $date;
121.           $status = 4;
122.           $cancel=0;
123.
124.           $result = $this->transaksiRepository-
       >findByTanggalPreOrderAndStatusAndTipeTransakis($ta
       nggal,$status,$tipe);
125.           $resultCount = $this->transaksiRepository-
       >getCountVerif($status,$tipe, $cancel);
126.           $tanggals=$this->transaksiRepository-
       >getTransactionDateWithStatus($status, $tipe, $canc
       el);
127.
128.           foreach($tanggals as $tgl){
```

```
129.                 $tgl->total=0;
130.                 foreach($resultCount as $resulttcount){

131.                     if($resulttcount->tanggal_pre_order==$tgl->tanggal_pre_order){
132.                         $tgl->total+=$resulttcount->total;
133.                     }
134.                     else{
135.                         $tgl->total+=0;
136.                     }
137.                 }
138.             }
139.
140.             $data = [
141.                 'preorder' => $result,
142.                 'isPaid' => [["Belum Lunas","0"],["Lunas","1"]],
143.                 'status' => [['Pre Order',1],['Siap Kirim',4],['Pengiriman',5],['Batal',7],['Selesai',6]],

144.                 'status_pembayaran' => [['Tunggu Konfirmasi',1],['Gagal',2],['Lunas',3],['Belum Lunas',0]],
145.                 'tanggal' => $tanggals,
146.                 'filter_tanggal' => $tanggal
147.             ];
148.
149.             return view('penjual.preorder_siap_kirim')->with('data',$data);
150.         }
151.
152.         public function dalamPengiriman($date){
153.
154.             $tipe = "FROM_BUYER";
155.             $tanggal = $date;
156.             $status = 6;
```

```
157.          $cancel=0;
158.
159.          $result = $this->transaksiRepository-
      >findByTanggalPreOrderAndStatusAndTipeTransakis($ta
      nggal,$status,$tipe);
160.          $resultCount = $this->transaksiRepository-
      >getCountVerif($status,$tipe, $cancel);
161.          $tanggals=$this->transaksiRepository-
      >getTransactionDateWithStatus($status, $tipe, $canc
      el);
162.
163.          // dd($result);
164.
165.          foreach($tanggals as $tgl){
166.              $tgl->total=0;
167.              foreach($resultCount as $resulttcount){

168.                  if($resulttcount-
      >tanggal_pre_order==$tgl->tanggal_pre_order){
169.                      $tgl->total+=$resulttcount-
      >total;
170.                  }
171.                  else{
172.                      $tgl->total+=0;
173.                  }
174.              }
175.          }
176.
177.          $data = [
178.              'preorder' => $result,
179.              'isPaid' => [["Belum Lunas","0"],["Luna
      s","1"]],
180.              'status' => [['Pre Order',1],['Siap Kir
      im',4],['Pengiriman',5],['Batal',7],['Selesai',6]],
```

```
181.                 'status_pembayaran' => [['Tunggu Konfir
masi',1],['Gagal',2],['Lunas',3],['Belum Lunas',0]]
,
182.                 'tanggal' => $tanggals,
183.                 'filter_tanggal' => $tanggal
184.             ];
185.
186.             return view('penjual.preorder_proses')-
>with('data',$data);
187.         }
188.
189.         public function selesai($date){
190.
191.             $tipe = "FROM_BUYER";
192.             $tanggal = $date;
193.             $status = 7;
194.             $cancel=0;
195.
196.             $result = $this->transaksiRepository-
>findByTanggalPreOrderAndStatusAndTipeTransakis($ta
nggal,$status,$tipe);
197.             $resultCount = $this->transaksiRepository-
>getCountVerif($status,$tipe, $cancel);
198.             $tanggals=$this->transaksiRepository-
>getTransactionDateWithStatus($status, $tipe, $canc
el);
199.
200.             foreach($tanggals as $tgl){
201.                 $tgl->total=0;
202.                 foreach($resultCount as $resulttcount){

203.                     if($resulttcount-
>tanggal_pre_order==$tgl->tanggal_pre_order){
204.                         $tgl->total+=$resulttcount-
>total;
205.                     }
206.                     else{
```

```
207.                     $tgl->total+=0;
208.                 }
209.             }
210.         }
211.
212.         $data = [
213.             'preorder' => $result,
214.             'isPaid' => [["Belum Lunas","0"],["Luna
    s","1"]],
215.             'status' => [['Pre Order',1],['Siap Kir
    im',4],['Pengiriman',5],['Batal',7],['Selesai',6]],

216.             'status_pembayaran' => [['Tunggu Konfir
    masi',1],['Gagal',2],['Lunas',3],['Belum Lunas',0]]
    ,
217.             'tanggal' => $tanggals,
218.             'filter_tanggal' => $tanggal
219.         ];
220.
221.         return view('penjual.preorder_selesai')-
    >with('data',$data);
222.     }
223.
224.     public function batal($date){
225.
226.         $tipe = "FROM_BUYER";
227.         $tanggal = $date;
228.         $cancel=1;
229.         $status=0;
230.
231.         $result = $this->transaksiRepository-
    >getAllTransaksiIsCancelled($tanggal,$tipe);
232.         $resultCount = $this->transaksiRepository-
    >getCountVerif($status,$tipe, $cancel);
233.         $tanggals=$this->transaksiRepository-
    >getTransactionDateWithStatus($status, $tipe, $canc
    el);
```

```
234.
235.           foreach($tanggals as $tgl){
236.               $tgl->total=0;
237.               foreach($resultCount as $resulttcount){
238.                   if($resulttcount->tanggal_pre_order==$tgl->tanggal_pre_order){
239.                       $tgl->total+=$resulttcount->total;
240.                   }
241.                   else{
242.                       $tgl->total+=0;
243.                   }
244.               }
245.           }
246.
247.           $data = [
248.               'preorder' => $result,
249.               'isPaid' => [["Belum Lunas","0"],["Lunas","1"]],
250.               'status' => [['Pre Order',1],['Siap Kirim',4],['Pengiriman',5],['Batal',7],['Selesai',6]],
251.               'status_pembayaran' => [['Tunggu Konfirmasi',1],['Gagal',2],['Lunas',3],['Belum Lunas',0]],
252.               'tanggal' => $tanggals,
253.               'filter_tanggal' => $tanggal
254.           ];
255.
256.           return view('penjual.preorder_batal')->with('data',$data);
257.       }
258.
259.       public function tambahPreOrder(){
260.           return view('penjual.tambah_preorder');
261.       }
```

```
262.
263.     public function _tambahPreOrder(){
264.
265.     }
266.
267.     public function _filterTanggal(Request $request
    ){
268.         session(['tanggal_pre_order' => $request-
    >input('tanggal')]);
269.         return redirect('/Penjual/PreOrder/Akumulas
    i');
270.     }
271.
272.     private function findObjectAkumulasi($kode_bara
    ng,$akumulasi){
273.         if(count($akumulasi) < 0){
274.             return false;
275.         }else{
276.             for($i=0;$i<count($akumulasi);$i++){
277.                 if($akumulasi[$i]-
    >kode_barang == $kode_barang){
278.                     return "key|".$i;
279.                 }
280.             }
281.         }
282.     }
283.
284.     public function rekap_akumulasi($tanggal){
285.         // dd($request->input());
286.
287.         $rekap = DB::select('call sp_get_total_pre_
    order(?, ?)', [$tanggal, Auth::user()->id]);
288.         $rekap_paket = DB::select('call sp_get_tota
    l_paket(?, ?)', [$tanggal, Auth::user()->id]);
289.         $rekap_non_paket = DB::select('call sp_get_
    total_non_paket(?, ?)', [$tanggal, Auth::user()-
    >id]);
```

```
290.          $rekap_timbang=DB::select('call sp_get_tota
      l_timbang(?, ?)', [$tanggal, Auth::user()->id]);
291.
292.          $merged = array_merge($rekap_paket, $rekap_
      non_paket, $rekap_timbang);
293.          $akumulasi = [];
294.
295.          $data = [
296.              'current_date' => $tanggal,
297.              'rekap' => $rekap,
298.              'rekap_sayur' => $merged
299.          ];
300.
301.          // dd($data);
302.
303.          return view('penjual.rekap_akumulasi')-
      >with('data',$data);
304.      }
305.
306.      public function rekap_detail($kode_barang,$date
      ){
307.          // $pembeli = DB::select('call sp_get_buyer
      _list(?,?)', [$kode_barang,$date]);
308.          // dd($pembeli);
309.          return $pembeli = DB::select('call sp_get_b
      uyer_list(?,?)', [$kode_barang,$date]);
310.      }
311.
312.      public function _rekap_akumulasi(Request $reque
      st){
313.
314.
315.          $tanggal = $request->input('date');
316.
317.          $rekap = DB::select('call sp_get_total_pre_
      order(?, ?)', [$tanggal, Auth::user()->id]);
```

```
318.          $rekap_paket = DB::select('call sp_get_tota
      l_paket(?, ?)', [$tanggal, Auth::user()->id]);
319.          $rekap_non_paket = DB::select('call sp_get_
      total_non_paket(?, ?)', [$tanggal, Auth::user()-
      >id]);
320.          $rekap_timbang=DB::select('call sp_get_tota
      l_timbang(?, ?)', [$tanggal, Auth::user()->id]);
321.
322.          $merged = array_merge($rekap_paket, $rekap_
      non_paket, $rekap_timbang);
323.          $akumulasi = [];
324.          $supplier = [];
325.
326.          // dd($merged);
327.
328.          // for($i = 0 ; $i<count($merged) ; $i++){

329.          //     $current = $merged[$i];
330.          //     if($this-
      >findObjectAkumulasi($current-
      >kode_barang, $akumulasi)){
331.          //         $temp = $this-
      >findObjectAkumulasi($current-
      >kode_barang, $akumulasi);
332.          //         $key = (int) explode("|", $temp)
      [1];
333.          //         $temp_total = (int) $akumulasi[$
      key]->total;
334.          //         $current_total = (int) $current-
      >total;
335.          //         $akumulasi[$key]-
      >total = $temp_total + $current_total;
336.          //     }else{
337.          //         array_push($akumulasi,$current);

338.          //     }
339.          // }
```

```
340.
341.           for($i = 0 ; $i<count($merged) ; $i++){
342.               $next = $merged[$i];
343.               if(in_array($next-
      >supplier_barang,$supplier)){
344.
345.               }else{
346.                   array_push($supplier,$next-
      >supplier_barang);
347.               }
348.           }
349.
350.           // dd([$supplier,$akumulasi]);
351.
352.           try {
353.               DB::beginTransaction();
354.               $list_transaksi = $this-
      >transaksiRepository->getIncludedRekap($request-
      >input('date'),0);
355.
356.               for($i = 0 ; $i<count($list_transaksi)
      ; $i++)
357.               {
358.                   $this->transaksiRepository-
      >update($list_transaksi[$i]->id,["status" => 2]);
359.                   $this->detailTransaksiRepository-
      >updateByIdTransaksi($list_transaksi[$i]-
      >id,["status" => 2]);
360.               }
361.
362.
363.               for($i = 0 ; $i<count($supplier) ; $i++
      ){
364.                   $petani = $this->userRepository-
      >find($supplier[$i]);
365.                   $kePetani = [
366.                       'id_user'=> $supplier[$i],
```

```
                        'petani' => $petani,
                        'tanggal' => $tanggal
                    ];
                    // dd($kePetani);
                    $transaksi = $this->transaksiRepository->orderKePetani($kePetani);
                    // dd($transaksi);
                    for($j = $i ; $j<count($merged) ; $j++){
                        if($merged[$j]->supplier_barang == $supplier[$i]){
                            // dd($akumulasi[$j]);
                            $barang = $this->produkRepository->findByKode($merged[$j]->kode_barang);
                            $detailKePetani = [
                                'id_transaksi' => $transaksi->id,
                                'barang' => $barang,
                                'akumulasi_barang' => $merged[$j]
                            ];
                            $this->detailTransaksiRepository->detailOrderKePetani($detailKePetani);
                        }
                    }
                }

                $list_transaksi = $this->transaksiRepository->getIncludedRekap($request->input('date'),1);
                for($i = 0 ; $i<count($list_transaksi) ; $i++){
                    $this->transaksiRepository->update($list_transaksi[$i]->id,["is_exclude_rekap" => 0]);
```

```
                    $this->detailTransaksiRepository->updateByIdTransaksi($list_transaksi[$i]->id,["is_exclude_rekap" => 0]);
                }

                DB::commit();
            } catch (\Throwable $th) {
                dd($th);
                DB::rollback();
            }

            return redirect('/Penjual/OrderPetani');
        }

        public function akumulasi(){
            $transcationDate = $this->transaksiRepository->getTransactionDateWithStatus(1, "FROM_BUYER", 0);

            if(count($transcationDate) == 1){
                $date = $transcationDate[0]->tanggal_pre_order;
                session(['tanggal_pre_order' => $date]);
            }
            elseif(session('tanggal_pre_order') == null){
                $date = Carbon::now()->toDateString();
                session(['tanggal_pre_order' => $transcationDate[0]->tanggal_pre_order]);
            }else{
                $date = session('tanggal_pre_order');
            }
            $status = 1;
            $tipe = "FROM_BUYER";
```

```
416.          $preordeer=$this->transaksiRepository->findByTanggalPreOrderAndStatusAndTipeTransakisAkumulasi($date,$status,$tipe);
417.          $data = [
418.              'preorder' => $preordeer,
419.              'current_date' => Carbon::now()->toDateString(),
420.              'date' => $transcationDate
421.          ];
422.
423.          // dd($data['preorder'][1]->detailTransaksi[0]);
424.          return view('penjual.tambah_akumulasi')->with('data',$data);
425.      }
426.
427.      public function _excludeDetailPreOrder(Request $request){
428.          try {
429.              DB::beginTransaction();
430.              $this->detailTransaksiRepository->excludeDetailPreOrder($request->id);
431.              DB::commit();
432.              return 1;
433.          } catch (\Throwable $th) {
434.              DB::rollback();
435.              return $th;
436.          }
437.      }
438.
439.      public function _excludePreOrder(Request $request){
440.          try {
441.              DB::beginTransaction();
442.              $details = $this->detailTransaksiRepository->findByTransaksiId($request->id);
```

```
443.                 $this->transaksiRepository->excludeRekapPreOrder($request->id);
444.                 foreach($details as $detail){
445.                     $this->detailTransaksiRepository->excludeDetailPreOrder($detail->id);
446.                 }
447.                 DB::commit();
448.                 return 1;
449.             } catch (\Throwable $th) {
450.                 DB::rollback();
451.                 return $th;
452.             }
453.         }
454.
455.         public function _batalkanDetailPreOrder(Request $request){
456.             try {
457.                 DB::beginTransaction();
458.                 $this->detailTransaksiRepository->cancelDetailPreOrder($request->id);
459.                 DB::commit();
460.                 return 1;
461.             } catch (\Throwable $th) {
462.                 DB::rollback();
463.                 return $th;
464.             }
465.         }
466.
467.         public function _batalkanPreOrder(Request $request){
468.             try {
469.                 DB::beginTransaction();
470.                 $details = $this->detailTransaksiRepository->findByTransaksiId($request->id);
471.                 $this->transaksiRepository->cancelPreOrder($request->id);
```

```
472.                 foreach($details as $detail){
473.                     $this->detailTransaksiRepository->cancelDetailPreOrder($detail->id);
474.                 }
475.                 DB::commit();
476.                 return 1;
477.             } catch (\Throwable $th) {
478.                 DB::rollback();
479.                 return $th;
480.             }
481.         }
482.
483.         public function _updatePembayaran(Request $request){
484.             // dd($request->input());
485.             try {
486.                 //code...
487.                 DB::begintransaction();
488.                     $id = $request->input('id_transaksi');
489.                     $data = [
490.                         'isAlreadyPay' => $request->input('status_transaksi')
491.                     ];
492.                     $this->transaksiRepository->update($id,$data);
493.                 DB::commit();
494.             } catch (\Throwable $th) {
495.                 //throw $th;
496.                 DB::rollback();
497.                 dd($th);
498.             }
499.
500.             return redirect()->back();
501.
502.
503.         }
```

```
504.      public function pembeliOffline($id){
505.          $pembeliOffline=$this->getPembeliOffline($id);
506.
507.          $data = [];
508.          foreach($pembeliOffline as $key => $off){
509.              $data[$key]['name'] = $off->name;
510.              $data[$key]['alamat'] = $off->alamat;
511.              $data[$key]['nohp'] = $off->nohp;
512.              // $data[$key]['data'] = $this->produkRepository->getEtalaseBarangByIdKategori($kategori->id);
513.              $data[$key]['data'] = $this->keranjangResellerRepository->getUserById($off->id);
514.
515.          }
516.
517.          $collection = collect($data);
518.          // dd($collection);
519.          return view('penjual.pembeli_offline')->with('collection',$collection);
520.      }
521.
522.      public function getPembeliOffline($id){
523.          $getParentKeranjang=$this->parentKeranjangResellerRepository->getFromIdTransaksi($id);
524.          return $getParentKeranjang;
525.      }
526.
527.      public function orderDetail($id){
528.          $getDetail=$this->detailTransaksiRepository->findByTransaksiId($id);
529.          $getTransaksi=$this->transaksiRepository->find($id);
```

```
530.            $data=[
531.                'detail'=>$getDetail,
532.                'transaksi'=>$getTransaksi
533.            ];
534.            // dd($collection);
535.            return view('penjual.detail_order')-
      >with('data',$data);
536.        }
537.    }
```

Kode Sumber 5.55 PreOrderController

5.3.2.10. ProdukController

Lapisan ini bertugas untuk mengatur Produk Penjual. Kode Sumber 5.56 berikut merupakan implementasi dari lapisan kontrol ProdukController:

```
1.    <?php
2.
3.    namespace App\Http\Controllers\Penjual;
4.
5.    use App\Http\Controllers\Controller;
6.    use Illuminate\Http\Request;
7.    use App\Repositories\ProdukRepository;
8.    use App\Repositories\UserRepository;
9.    use App\Repositories\KategoriRepository;
10.   use App\Repositories\BaseKategoriRepository;
11.   use App\Repositories\BobotKemasanRepository;
12.   use App\Repositories\ProdukKemasanRepository;
13.   use App\Repositories\ProdukGroupRepository;
14.   use App\Repositories\FotoProdukRepository;
15.   use App\Repositories\InventarisRepository;
16.   use App\Repositories\DetailTransaksiRepository;
17.   use App\Repositories\DetailKeranjangRepository;
18.   use App\Repositories\DetailKlaimRepository;
19.   use App\Repositories\KeranjangResellerRepository;
```

```
20.    use App\Repositories\BarangTanggalRepository;
21.    use App\Repositories\IsiPaketRepository;
22.    use App\Repositories\IsiResepRepository;
23.
24.    use Illuminate\Support\Facades\Storage;
25.    use Response;
26.    use DB;
27.    use Auth;
28.
29.
30.    class ProdukController extends Controller
31.    {
32.
33.        protected $produkRepository, $userRepository, $
       kategoriRepository, $bobotKemasanRepository, $baseK
       ategoriRepository, $produkKemasanRepository, $produ
       kGroupRepository, $fotoProdukRepository, $inventari
       sRepository, $detailTransaksiRepository, $detailKer
       anjangRepository,$detailKlaimRepository, $keranjang
       ResellerRepository, $barangTanggalRepository, $isiP
       aketRepository, $isiResepRepository;
34.
35.        public function __construct(ProdukRepository $p
       rodukRepository, UserRepository $userRepository, Ka
       tegoriRepository $kategoriRepository, BobotKemasanR
       epository $bobotKemasanRepository, BaseKategoriRepo
       sitory $baseKategoriRepository, ProdukKemasanReposi
       tory $produkKemasanRepository, ProdukGroupReposito
       y $produkGroupRepository, FotoProdukRepository $fot
       oProdukRepository, InventarisRepository $inventaris
       Repository, DetailTransaksiRepository $detailTransa
       ksiRepository, DetailKeranjangRepository $detailKer
       anjangRepository,DetailKlaimRepository $detailKlaim
       Repository, KeranjangResellerRepository $keranjangR
       esellerRepository,BarangTanggalRepository $barangTa
       nggalRepository,IsiPaketRepository $isiPaketReposit
       ory,IsiResepRepository $isiResepRepository)
```

```
36.         {
37.             $this->produkRepository = $produkRepository;
38.             $this->userRepository = $userRepository;
39.             $this->kategoriRepository = $kategoriRepository;
40.             $this->bobotKemasanRepository = $bobotKemasanRepository;
41.             $this->baseKategoriRepository = $baseKategoriRepository;
42.             $this->produkKemasanRepository = $produkKemasanRepository;
43.             $this->produkGroupRepository = $produkGroupRepository;
44.             $this->fotoProdukRepository = $fotoProdukRepository;
45.             $this->inventarisRepository = $inventarisRepository;
46.             $this->detailTransaksiRepository=$detailTransaksiRepository;
47.             $this->detailKeranjangRepository=$detailKeranjangRepository;
48.             $this->detailKlaimRepository=$detailKlaimRepository;
49.             $this->keranjangResellerRepository= $keranjangResellerRepository;
50.             $this->barangTanggalRepository=$barangTanggalRepository;
51.             $this->isiPaketRepository=$isiPaketRepository;
```

```
52.             $this->isiResepRepository=$isiResepRepository;
53.
54.
55.             $this->middleware('auth');
56.             $this->middleware('penjual');
57.
58.         }
59.
60.         public function listProduk()
61.         {
62.             $getAllProduk = $this->produkRepository->getProduct();
63.
64.             return view('penjual.produk')->with('produks', $getAllProduk);
65.         }
66.
67.         public function tambahProduk()
68.         {
69.
70.             $getAllSupplier = $this->userRepository->getPetani();
71.             $kodebarang = $this->generateKode();
72.             $subKategori = $this->kategoriRepository->all();
73.             $bobotKemasan = $this->bobotKemasanRepository->all();
74.             $baseKategori = $this->baseKategoriRepository->all();
75.
76.             $data = [
77.                 'supplier' => $getAllSupplier,
78.                 'kodebarang' => $kodebarang,
79.                 'group' => $subKategori,
80.                 'kemasan_value' => $bobotKemasan,
81.                 'kategori_value' => $baseKategori,
```

```
82.                 'jenis_value' => ['Kemas', 'Timbang'],
83.                 'satuan_value' => ['Gram','Pcs']
84.             ];
85.
86.             return view('penjual.tambah_produk')->with('data', $data);
87.         }
88.
89.         public function _tambahProduk(Request $request)
90.         {
91.             if(count($request->file('foto_barang')) > 5){
92.                 return redirect()->back()->with('error',"Maksimal Upload 5 Foto!")->withInput();
93.             }
94.
95.             session(['id_supplier' => $request->input('id_user')]);
96.             if($request->jenis_diskon=="Potongan Persen"){
97.                 if ($request->jenis_diskon_reseller=="Potongan Persen") {
98.                     $harga_diskon=$request->harga_jual-($request->harga_jual*($request->diskon/100));
99.                     $reseller=($harga_diskon-($harga_diskon*($request->diskon_reseller/100)));
100.                } else {
101.                    $harga_diskon=$request->harga_jual-($request->harga_jual*($request->diskon/100));
102.                    $reseller=$harga_diskon-$request->diskon_reseller;
103.                }
104.            }
105.            else{
```

```
106.                if ($request->jenis_diskon_reseller=="Potongan Persen") {
107.                    $harga_diskon=$request->harga_jual-$request->diskon;
108.                    $reseller=($harga_diskon-($harga_diskon*($request->diskon_reseller/100)));
109.                } else {
110.                    $harga_diskon=$request->harga_jual-$request->diskon;
111.                    $reseller=$harga_diskon-$request->diskon_reseller;
112.                }
113.            }
114.
115.            $request->merge([
116.                'harga_jual_reseller' => $reseller,
117.            ]);
118.
119.            DB::beginTransaction();
120.            try {
121.                $this->produkRepository->create($request->all());
122.                if($request->input('bobot_kemasan') !== null){
123.                    $this->produkKemasanRepository->create($request->input('id'), $request->input('bobot_kemasan'));
124.                }
125.                $this->produkGroupRepository->create($request->input('id'), $request->input('group_etalase'));
126.                $this->uploadPhoto($request->input('id'),$request->file('foto_barang'));
127.
128.                DB::commit();
```

```
129.                 return redirect()->back()->with('success',"Berhasil Menambah!");
130.             } catch (\Throwable $th) {
131.                 DB::rollback();
132.                 return redirect()->back()->with('error',"Gagal menambahkan barang!")->withInput();
133.
134.             }
135.
136.             return redirect()->back();
137.         }
138.
139.         private function generateKode(){
140.
141.             $finalStr = 'B'.rand(1000,9999);
142.
143.             $kode = $this->produkRepository->findByKode($finalStr);
144.             if($kode){
145.                 $this->generateKode();
146.             }else{
147.                 return $finalStr;
148.             }
149.         }
150.
151.         private function uploadPhoto($id,$datas){
152.
153.             $fotos = $this->fotoProdukRepository->findByIdBarang($id);
154.
155.             foreach($fotos as $foto){
156.                 $path = public_path().'\img\foto_barang\\';
157.                 $filename = $path.$foto->path;
158.                 if(file_exists($filename)){
```

```
159.                     unlink($filename);
160.                     $this->fotoProdukRepository->delete($id,$foto->path);
161.                 }
162.             }
163.
164.
165.             foreach($datas as $data){
166.                 $filename = time().$data->getClientOriginalName();
167.                 $path = public_path().'/img/foto_barang';
168.                 $data->move($path,$filename);
169.                 $this->fotoProdukRepository->create($id, $filename);
170.             }
171.
172.         }
173.
174.         public function index()
175.         {
176.             return view('penjual.home');
177.         }
178.
179.         public function ubahProduk($id){
180.
181.             $getAllSupplier = $this->userRepository->getPetani();
182.             $getBarang = $this->produkRepository->findById($id);
183.             $subKategori = $this->kategoriRepository->all();
184.             $barangKemasan = $this->produkKemasanRepository->findByIdBarang($id);
185.             $barangGroup = $this->produkGroupRepository->findByIdBarang($id);
```

```
186.          $bobotKemasan = $this-
      >bobotKemasanRepository->all();
187.          $baseKategori = $this-
      >baseKategoriRepository->all();
188.
189.
190.          for($i=0;$i<count($bobotKemasan);$i++){
191.              for($j=0;$j<count($barangKemasan);$j++)
      {
192.                  if(($bobotKemasan[$i]-
      >bobot_kemasan == $barangKemasan[$j]-
      >bobot_kemasan) && ($bobotKemasan[$i]-
      >selected != 1)){
193.                      $bobotKemasan[$i]-
      >selected = 1;
194.                  }elseif($bobotKemasan[$i]-
      >selected == 0){
195.                       $bobotKemasan[$i]-
      >selected = 0;
196.                  }
197.              }
198.          }
199.
200.          for($i=0;$i<count($subKategori);$i++){
201.              for($j=0;$j<count($barangGroup);$j++){

202.                  if(($subKategori[$i]-
      >id == $barangGroup[$j]-
      >id_kategori) && ($subKategori[$i]-
      >selected != 1)){
203.                      $subKategori[$i]-
      >selected = 1;
204.                  }elseif($subKategori[$i]-
      >selected == 0){
205.                       $subKategori[$i]-
      >selected = 0;
206.                  }
```

```
207.                 }
208.             }
209.
210.             $data = [
211.                 'supplier' => $getAllSupplier,
212.                 'barang' => $getBarang,
213.                 'group' => $subKategori,
214.                 'kemasan_barang' => $barangKemasan,
215.                 'group_barang' => $barangGroup,
216.                 'kemasan_value' => $bobotKemasan,
217.                 'kategori_value' => $baseKategori,
218.                 'jenis_value' => ['Kemas', 'Timbang'],

219.                 'satuan_value' => ['Gram','Pcs']
220.             ];
221.
222.             return view('penjual.ubah_produk')-
    >with('data',$data);
223.         }
224.
225.         public function _ubahProduk(Request $request){

226.
227.             session(['id_supplier' => $request-
    >input('id_user')]);
228.
229.             if($request-
    >jenis_diskon=="Potongan Persen"){
230.                 if ($request-
    >jenis_diskon_reseller=="Potongan Persen") {
231.                     $harga_diskon=$request->harga_jual-
    ($request->harga_jual*($request->diskon/100));
232.                     $reseller=($harga_diskon-
    ($harga_diskon*($request->diskon_reseller/100)));
233.                 } else {
234.                     $harga_diskon=$request->harga_jual-
    ($request->harga_jual*($request->diskon/100));
```

```
235.                 $reseller=$harga_diskon-$request->diskon_reseller;
236.             }
237.         }
238.         else{
239.             if ($request->jenis_diskon_reseller=="Potongan Persen") {
240.                 $harga_diskon=$request->harga_jual-$request->diskon;
241.                 $reseller=($harga_diskon-($harga_diskon*($request->diskon_reseller/100)));
242.             } else {
243.                 $harga_diskon=$request->harga_jual-$request->diskon;
244.                 $reseller=$harga_diskon-$request->diskon_reseller;
245.             }
246.         }
247.
248.         $request->merge([
249.             'harga_jual_reseller' => $reseller,
250.         ]);
251.
252.         DB::beginTransaction();
253.         try {
254.             $this->produkGroupRepository->delete($request->id);
255.             $this->produkKemasanRepository->delete($request->id);
256.             $this->produkRepository->update($request->id,$request->all());
257.
258.             if($request->input('bobot_kemasan') !== null){
259.                 $this->produkKemasanRepository->create($request->input('id'), $request->input('bobot_kemasan'));
```

```
260.                 }
261.                 $this->produkGroupRepository->create($request->input('id'), $request->input('group_etalase'));
262.
263.
264.                 DB::commit();
265.                 return redirect()->back()->with('success',"Berhasil Mengubah!")->withInput();
266.
267.             } catch (\Throwable $th) {
268.                 DB::rollback();
269.                 dd($th);
270.                 return redirect()->back()->with('error',"Gagal menambahkan barang!")->withInput();
271.
272.             }
273.
274.             return redirect()->back();
275.
276.         }
277.
278.         public function inventarisProduk(){
279.
280.             $data = [
281.                 'inventaris' => $this->produkRepository->getProduct(),
282.                 'keluar_masuk' => $this->inventarisRepository->all(),
283.                 'nama'=>"Produk"
284.             ];
285.
286.             // dd($data);
287.
```

```
288.            return view('penjual.inventaris')-
    >with('data',$data);
289.
290.        }
291.        public function inventarisPaket(){
292.
293.            $data = [
294.                'inventaris' => $this-
    >produkRepository->getPaket(),
295.                'keluar_masuk' => $this-
    >inventarisRepository->all(),
296.                'nama'=>"Paket"
297.            ];
298.
299.            // dd($data);
300.
301.            return view('penjual.inventaris')-
    >with('data',$data);
302.
303.        }
304.
305.        public function tambahInventaris(){
306.            $getAllSupplier = $this->userRepository-
    >getPetani();
307.            $kodebarang = $this->generateKode();
308.            $subKategori = $this->kategoriRepository-
    >all();
309.            $bobotKemasan = $this-
    >bobotKemasanRepository->all();
310.            $baseKategori = $this-
    >baseKategoriRepository->all();
311.
312.            $data = [
313.                'supplier' => $getAllSupplier,
314.                'kodebarang' => $kodebarang,
315.                'group' => $subKategori,
316.                'kemasan_value' => $bobotKemasan,
```

```
317.                'kategori_value' => $baseKategori,
318.                'jenis_value' => ['Kemas', 'Timbang'],
319.                'satuan_value' => ['Gram','Pcs']
320.            ];
321.
322.            return view('penjual.tambah_inventaris')->with('data',$data);
323.        }
324.
325.        public function _tambahInventaris(Request $request){
326.
327.        }
328.
329.        public function ubahInventaris($id){
330.            $getAllSupplier = $this->userRepository->getPetani();
331.            $getBarang = $this->produkRepository->findById($id);
332.            $subKategori = $this->kategoriRepository->all();
333.            $barangKemasan = $this->produkKemasanRepository->findByIdBarang($id);
334.            $barangGroup = $this->produkGroupRepository->findByIdBarang($id);
335.            $bobotKemasan = $this->bobotKemasanRepository->all();
336.            $baseKategori = $this->baseKategoriRepository->all();
337.
338.
339.            for($i=0;$i<count($bobotKemasan);$i++){
340.                for($j=0;$j<count($barangKemasan);$j++){
341.                    if(($bobotKemasan[$i]->bobot_kemasan == $barangKemasan[$j]->
```

```
>bobot_kemasan) && ($bobotKemasan[$i]-
>selected != 1)){
342.                        $bobotKemasan[$i]-
>selected = 1;
343.                    }elseif($bobotKemasan[$i]-
>selected == 0){
344.                         $bobotKemasan[$i]-
>selected = 0;
345.                    }
346.                }
347.            }
348.
349.            for($i=0;$i<count($subKategori);$i++){
350.                for($j=0;$j<count($barangGroup);$j++){

351.                    if(($subKategori[$i]-
>id == $barangGroup[$j]-
>id_kategori) && ($subKategori[$i]-
>selected != 1)){
352.                        $subKategori[$i]-
>selected = 1;
353.                    }elseif($subKategori[$i]-
>selected == 0){
354.                         $subKategori[$i]-
>selected = 0;
355.                    }
356.                }
357.            }
358.
359.            $data = [
360.                'supplier' => $getAllSupplier,
361.                'barang' => $getBarang,
362.                'group' => $subKategori,
363.                'kemasan_barang' => $barangKemasan,
364.                'group_barang' => $barangGroup,
365.                'kemasan_value' => $bobotKemasan,
366.                'kategori_value' => $baseKategori,
```

```
367.                'jenis_value' => ['Kemas', 'Timbang'],
368.                'satuan_value' => ['Gram','Pcs']
369.            ];
370.
371.
372.            return view('penjual.ubah_inventaris')->with('data',$data);
373.        }
374.
375.        public function _ubahInventaris(Request $request){
376.            // dd($request->input());
377.            $id_barang = $request->input('id');
378.            $stok = $request->input('stok')+$request->input('stok_awal');
379.            $stok_manual=$request->input('stok');
380.            $keterangan = "Tambah Stok";
381.            $this->produkRepository->update($id_barang,['stok' => $stok]);
382.            $this->inventarisRepository->inventarisIn($id_barang,'tambah manual',$stok_manual,$keterangan);
383.            return redirect('/Penjual/Inventaris/Produk');
384.
385.        }
386.
387.        public function stok(){
388.
389.            $barang = $this->produkRepository->findByUserId(Auth::user()->id);
390.
391.            $data = [
392.                'barang' => $barang
393.            ];
```

```
394.            return view('penjual.stok')-
    >with('data',$data);
395.        }
396.
397.        public function _stok(Request $request){
398.
399.
400.            $size = count($request->input());
401.            for ($i=0; $i < $size; $i++) {
402.
403.                $index_id_barang = 'id_barang_'.$i;
404.                $index_ketersediaan = 'ketersediaan_'.$
    i;
405.
406.                $id = $request-
    >input($index_id_barang);
407.                $barang = ['ketersediaan' => $request-
    >input($index_ketersediaan)];
408.
409.                try {
410.                    DB::beginTransaction();
411.                    $this->produkRepository-
    >update($id,$barang);
412.                    DB::commit();
413.                } catch (\Throwable $th) {
414.                    DB::rollback();
415.                }
416.
417.            }
418.
419.            return redirect()->back();
420.        }
421.
422.        public function _hapusProduk(Request $request){

423.            $id = $request->id;
424.            try {
```

```
425.                 DB::beginTransaction();
426.                 $this->produkRepository->delete($id);
427.                 $this->produkKemasanRepository-
    >deleteByIdBarang($id);
428.                 $this->produkGroupRepository-
    >deleteByIdBarang($id);
429.                 $this->fotoProdukRepository-
    >deleteByIdBarang($id);
430.                 $this->inventarisRepository-
    >deleteByIdBarang($id);
431.                 $this->detailTransaksiRepository-
    >deleteByIdBarang($id);
432.                 $this->detailKeranjangRepository-
    >deleteByIdBarang($id);
433.                 $this->detailKlaimRepository-
    >deleteByIdBarang($id);
434.                 $this->keranjangResellerRepository-
    >deleteByIdBarang($id);
435.                 $this->barangTanggalRepository-
    >deleteByIdBarang($id);
436.                 $this->isiPaketRepository-
    >deleteByIdBarang($id);
437.                 $this->isiResepRepository-
    >deleteByIdBarang($id);
438.                 $this->isiPaketRepository-
    >deleteByIdParentBarang($id);
439.                 $this->isiResepRepository-
    >deleteByIdParentBarang($id);
440.                 DB::commit();
441.             } catch (\Throwable $th) {
442.                 DB::rollback();
443.             }
444.             // $this->produkRepository->delete($id);
445.             return redirect()->back();
446.         }
447.
448.     }
```

Kode Sumber 5.56 ProdukController

5.3.2.11. ReportController

Lapisan ini bertugas untuk mengatur Laporan Penjual. Kode Sumber 5.57 berikut merupakan implementasi dari lapisan kontrol ReportController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Penjual;
4.
5.  use App\Http\Controllers\Controller;
6.  use Illuminate\Http\Request;
7.  use App\Repositories\TransaksiRepository;
8.
9.  use Illuminate\Support\Facades\Storage;
10. use Response;
11. use DB;
12.
13.
14. class ReportController extends Controller
15. {
16.
17.     protected $transaksiRepository;
18.
19.     public function __construct(TransaksiRepository $
    transaksiRepository)
20.     {
21.         $this-
    >transaksiRepository = $transaksiRepository;
22.
23.         $this->middleware('auth');
24.         $this->middleware('penjual');
25.
26.     }
27.
```

```
28.     public function reportHarian($date){
29.         $getPemasukan=DB::select('call sp_get_pemasuk
    an_pertgl(?)',array($date));
30.         $getPengeluaran=DB::select('call sp_get_penge
    luaran_pertgl(?)',array($date));
31.         $getPemasukanTotal=$this-
    >transaksiRepository-
    >getPemasukanTotalHarian($date);
32.         $getPengeluaranTotal=DB::select('call sp_get_
    total_bayar_veggo_by_petani(?)',array($date));
33.         // dd($getPengeluaranTotal);
34.         $pemasukanTotalInt=(int)$getPemasukanTotal-
    >pemasukan;
35.         if(count($getPengeluaranTotal)>0){
36.             $pengeluaranTotalInt=(int)$getPengeluaran
    Total[0]->harga;
37.         }
38.         else{
39.             $pengeluaranTotalInt=0;
40.         }
41.
42.         // dd($pengeluaranTotalInt);
43.         $labaTotal=$pemasukanTotalInt-
    $pengeluaranTotalInt;
44.         $hasil=array();
45.         $hasil2=array();
46.         $flag=0;
47.
48.         foreach($getPemasukan as $key => $masuk){
49.             $hasil[$key]['nama']= $masuk->nama;
50.             $hasil[$key]['pemasukan']= $masuk-
    >pemasukan;
51.             $hasil[$key]['pengeluaran']= 0;
52.             $hasil[$key]['laba']= $hasil[$key]['pemas
    ukan'] - $hasil[$key]['pengeluaran'] ;
53.         }
```

```
54.         foreach($getPengeluaran as $key => $keluar){
55.             foreach($hasil as $key=> $hasill){
56.                 if($hasil[$key]['nama']==$keluar->nama){
57.                     $hasil[$key]['pengeluaran']=$keluar->pengeluaran;
58.                     $hasil[$key]['laba']=$hasil[$key]['pemasukan']-$hasil[$key]['pengeluaran'];
59.                     $flag=1;
60.                     break;
61.                 }
62.             }
63.             if($flag==0){
64.                 $hasil2[$key]['nama']= $keluar->nama;
65.                 $hasil2[$key]['pemasukan']= 0;
66.                 $hasil2[$key]['pengeluaran']= $keluar->pengeluaran;
67.                 $hasil2[$key]['laba']=$hasil2[$key]['pemasukan']-$hasil2[$key]['pengeluaran'];
68.             }
69.             $flag=0;
70.         }
71.         $data=
72.         [
73.             'hasil'=>array_merge($hasil, $hasil2),
74.             'tanggal'=>$this->transaksiRepository->getTgl(),
75.             'filter_tanggal' =>$date,
76.             'pengeluaranTotal'=>$pengeluaranTotalInt,
77.             'pemasukanTotal'=>$pemasukanTotalInt,
78.             'labaTotal'=>$labaTotal
79.         ];
80.         // dd($data['hasil'][0]['nama']);
81.         // $produk=;
```

```
82.
83.         return view('penjual.report_harian')-
    >with(compact('data'));
84.     }
85.
86.     public function reportBulanan($bulan, $tahun){
87.         $getPemasukan=DB::select('call sp_get_pemasuk
    an_perbln(?, ?)',array($bulan, $tahun));
88.         $getPengeluaran=DB::select('call sp_get_penge
    luaran_perbln(?, ?)',array($bulan, $tahun));
89.         $getPemasukanTotal=$this-
    >transaksiRepository-
    >getPemasukanTotalBulanan($bulan, $tahun);
90.         // dd($getPemasukanTotal);
91.         $getPengeluaranTotal=DB::select('call sp_get_
    pengeluaran_total_perbln(?, ?)',array($bulan, $tahun)
    );
92.         // dd($getPemasukanTotal->pemasukan);
93.         // dd($getPengeluaranTotal);
94.         $pemasukanTotalInt=(int)$getPemasukanTotal-
    >pemasukan;
95.         $pengeluaranTotalInt=(int)$getPengeluaranTota
    l[0]->pengeluaran;
96.         // dd($pengeluaranTotalInt);
97.         $labaTotal=$pemasukanTotalInt-
    $pengeluaranTotalInt;
98.         $hasil=array();
99.         $hasil2=array();
100.        $flag=0;
101.
102.        foreach($getPemasukan as $key => $masuk){
103.            $hasil[$key]['nama']= $masuk->nama;
104.            $hasil[$key]['pemasukan']= $masuk-
    >pemasukan;
105.            $hasil[$key]['pengeluaran']= 0;
106.            $hasil[$key]['laba']= $hasil[$key]['pemas
    ukan'] - $hasil[$key]['pengeluaran'] ;
```

```
107.         }
108.         foreach($getPengeluaran as $key => $keluar){

109.             foreach($hasil as $key=> $hasill){
110.                 if($hasil[$key]['nama']==$keluar-
    >nama){
111.                     $hasil[$key]['pengeluaran']=$kelu
    ar->pengeluaran;
112.                     $hasil[$key]['laba']=$hasil[$key]
    ['pemasukan']-$hasil[$key]['pengeluaran'];
113.                     $flag=1;
114.                     break;
115.                 }
116.             }
117.             if($flag==0){
118.                 $hasil2[$key]['nama']= $keluar-
    >nama;
119.                 $hasil2[$key]['pemasukan']= 0;
120.                 $hasil2[$key]['pengeluaran']= $keluar
    ->pengeluaran;
121.                 $hasil2[$key]['laba']=$hasil2[$key]['
    pemasukan']-$hasil2[$key]['pengeluaran'];
122.             }
123.             $flag=0;
124.         }
125.         $data=
126.         [
127.             'hasil'=>array_merge($hasil, $hasil2),
128.             'tanggal'=>$this->transaksiRepository-
    >getBulanTahun(),
129.             'bulan' =>date('F', mktime(0, 0, 0, $bula
    n, 10)),
130.             'bulanint'=>$bulan,
131.             'tahun'=>$tahun,
132.             'pengeluaranTotal'=>$pengeluaranTotalInt,

133.             'pemasukanTotal'=>$pemasukanTotalInt,
```

```
134.             'labaTotal'=>$labaTotal
135.         ];
136.         foreach($data['tanggal'] as $tanggal){
137.             // dd($tanggal->bulan);
138.             $tanggal->bulanNama = date('F', mktime(0, 0, 0, $tanggal->bulan, 10));
139.         }
140.
141.         // dd($data['tanggal']);
142.
143.         return view('penjual.report_bulanan')->with('data',$data);
144.     }
145.
146.     public function reportReseller($date){
147.         $reseller=$this->transaksiRepository->getResellerBydate($date);
148.         $data=[
149.             'tanggal'=>$this->transaksiRepository->getTgl(),
150.             'reseller'=>$reseller,
151.             'filter_tanggal'=> $date
152.         ];
153.
154.         // dd($data['reseller']);
155.
156.         return  view('penjual.report_reseller')->with('data', $data);
157.
158.     }
159.
160.     public function reportResellerDetail($date, $id){

161.         $kemas=DB::select('call sp_get_report_reseller_by_date_and_kemas(?, ?)',array($date, $id));
```

```
162.         $paket=DB::select('call sp_get_report_reselle
     r_by_date_and_paket(?,?)',array($date, $id));
163.         $timbang=DB::select('call sp_get_report_resel
     ler_by_date_and_timbang(?,?)',array($date, $id));
164.         $total=DB::select('call sp_get_report_reselle
     r_total(?,?)',array($date, $id));
165.
166.         $data=[
167.             'kemas'=>$kemas,
168.             'timbang'=>$timbang,
169.             'paket'=>$paket,
170.             'total'=>$total,
171.         ];
172.
173.         return view('penjual.report_reseller_detail')
     ->with('data', $data);
174.
175.     }
176.
177. }
```

Kode Sumber 5.57 ReportController

5.3.3. Role Petani

merupakan kumpulan *Controller* yang dikhususkan untuk role Petani. Berikut adalah beberapa *Controller*:

5.3.3.1. HomeController

Lapisan ini bertugas untuk mengatur *Home* Petani. Kode Sumber 5.58 berikut merupakan implementasi dari lapisan kontrol HomeController:

```
1.  <?php
2.
```

```
3.  namespace App\Http\Controllers\Petani;
4.
5.  use App\Http\Controllers\Controller;
6.  use Illuminate\Http\Request;
7.  use Illuminate\Support\Facades\Auth;
8.  use App\Repositories\TransaksiRepository;
9.  use App\Repositories\DetailTransaksiRepository;
10. use App\Repositories\ProdukRepository;
11. use DB;
12. use Carbon\Carbon;
13.
14. class HomeController extends Controller
15. {
16.     private $transaksiRepository,
17.             $detailTransaksiRepository,
18.             $produkRepository;
19.
20.     public function __construct(TransaksiRepository
    $transaksiRepository, DetailTransaksiRepository $de
    tailTransaksiRepository, ProdukRepository $produkRe
    pository)
21.     {
22.         $this-
    >transaksiRepository = $transaksiRepository;
23.         $this-
    >detailTransaksiRepository = $detailTransaksiReposi
    tory;
24.         $this-
    >produkRepository = $produkRepository;
25.
26.         $this->middleware('auth');
27.         $this->middleware('petani');
28.     }
29.
30.     public function index()
31.     {
32.         return view('petani.home');
```

```
33.         }
34.
35.         public function demand($date)
36.         {
37.
38.             $data = [
39.                 'demand' => "",
40.                 'selected' => $date,
41.                 'tanggal' => $transcationDate = $this-
    >transaksiRepository->getTransactionDate(),
42.             ];
43.
44.             return view('petani.demand')-
    >with('data',$data);
45.         }
46.
47.         public function order(){
48.
49.             $data = [
50.                 'order' => $this->transaksiRepository-
    >findByIdUser(Auth::user()->id)
51.             ];
52.
53.             return view('petani.order')-
    >with('data',$data);
54.         }
55.
56.         public function orderDetail($id){
57.             return "order:".$id;
58.         }
59.
60.         public function stok(){
61.
62.             $barang = $this->produkRepository-
    >findByUserId(Auth::user()->id);
63.
64.             $data = [
```

```
65.                 'barang' => $barang
66.             ];
67.             return view('petani.stok')-
    >with('data',$data);
68.         }
69.
70.         public function _stok(Request $request){
71.
72.
73.             $size = count($request->input());
74.             for ($i=0; $i < $size; $i++) {
75.
76.                 $index_id_barang = 'id_barang_'.$i;
77.                 $index_ketersediaan = 'ketersediaan_'.$
    i;
78.
79.                 $id = $request-
    >input($index_id_barang);
80.                 $barang = ['ketersediaan' => $request-
    >input($index_ketersediaan)];
81.
82.                 try {
83.                     DB::beginTransaction();
84.                     $this->produkRepository-
    >update($id,$barang);
85.                     DB::commit();
86.                 } catch (\Throwable $th) {
87.                     DB::rollback();
88.                 }
89.
90.             }
91.
92.             return redirect()->back();
93.         }
94.
95.         public function orderKonfirmasi($id){
```

```
96.             $transaksi = $this->transaksiRepository-
    >find($id);
97.             $data = [
98.                 'order' => $transaksi
99.             ];
100.            return view('petani.konfirmasi_order')-
    >with('data',$data);
101.
102.        }
103.
104.        public function _orderKonfirmasi(Request $reque
    st){
105.            // dd($request->input());
106.            $detailTransaksi = $request-
    >input('id_detail_transaksi');
107.            $volumeKirim = $request-
    >input('volume_kirim');
108.            $bobotKirim = $request-
    >input('bobot_kirim');
109.            $selisihKirim=$request-
    >input('selisih_kirim');
110.            $keterangan = $request-
    >input('keterangan');
111.            // dd($keterangan);
112.            try {
113.                DB::beginTransaction();
114.                $flag = 2;
115.                $this->transaksiRepository-
    >updateStatusOrderKePetani($request-
    >input('id_transaksi'),$flag);
116.                $this->transaksiRepository-
    >updateTanggalPengiriman($request-
    >input('id_transaksi'),Carbon::now());
117.                foreach ($detailTransaksi as $key => $v
    alue) {
118.                    $id = $detailTransaksi[$key];
119.                    $value = $volumeKirim[$key];
```

```
                    $value2=$bobotKirim[$key];
                    $value3=$selisihKirim[$key];
                    // dd($keterangan[$key]);
                    $keterangans = $keterangan[$key];
                    $this->detailTransaksiRepository->updatePengirimanPetani($id,$value, $value2,$value3, $keterangans,$flag);
                }
                DB::commit();
            } catch (\Throwable $th) {
                DB::rollback();
                dd($th);
            }

            return redirect('/Petani/Order');
        }
    }
```

Kode Sumber 5.58 HomeController

5.3.3.2. KlaimController

Lapisan ini bertugas untuk mengatur Klaim Petani. Kode Sumber 5.59 berikut merupakan implementasi dari lapisan kontrol KlaimController:

```
<?php

namespace App\Http\Controllers\Petani;

use App\Http\Controllers\Controller;
use App\Repositories\TransaksiRepository;
use App\Repositories\ProdukRepository;
use App\Repositories\KlaimRepository;
use App\Repositories\DetailKlaimRepository;
use Illuminate\Http\Request;
use \stdClass;
```

```
12.  use DB;
13.  use Auth;
14.
15.  class KlaimController extends Controller
16.  {
17.      //
18.      protected $transaksiRepository,
19.                $klaimRepository,
20.                $detailKlaimRepository,
21.                $produkRepository;
22.
23.
24.      public function __construct(TransaksiRepository
     $transaksiRepository, ProdukRepository $produkRepos
     itory,KlaimRepository $klaimRepository,DetailKlaimR
     epository $detailKlaimRepository)
25.      {
26.          $this-
     >transaksiRepository = $transaksiRepository;
27.          $this-
     >produkRepository = $produkRepository;
28.          $this-
     >klaimRepository = $klaimRepository;
29.          $this-
     >detailKlaimRepository = $detailKlaimRepository;
30.          $this->middleware('auth');
31.          $this->middleware('petani');
32.      }
33.
34.      public function klaim(){
35.          $data = [
36.              'klaim' => $this->klaimRepository-
     >getById(Auth::user()->id)
37.          ];
38.
39.          // dd($data['klaim']->dari);
40.
```

```
41.             return view('petani.klaim')-
    >with('data',$data);
42.
43.         }
44.
45.         public function detailKlaim($id){
46.             $data = [
47.                 'klaim' => $this-
    >detailKlaimRepository->getDetailKlaimById($id)
48.             ];
49.             // dd($data['klaim']);
50.
51.             return view('petani.detail_klaim')-
    >with('data', $data);
52.         }
53.     }
```

Kode Sumber 5.59 KlaimController

5.3.3.3. PembayaranController

Lapisan ini bertugas untuk mengatur Pembayaran Petani. Kode Sumber 5.60 berikut merupakan implementasi dari lapisan kontrol PembayaranController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Petani;
4.
5.  use App\Http\Controllers\Controller;
6.  use Illuminate\Http\Request;
7.  use App\Repositories\TransaksiRepository;
8.
9.  use Illuminate\Support\Facades\Storage;
10. use Response;
11. use DB;
12.
```

```
13.
14. class PembayaranController extends Controller
15. {
16.
17.     protected $transaksiRepository;
18.
19.     public function __construct(TransaksiRepository $
    transaksiRepository)
20.     {
21.         $this-
    >transaksiRepository = $transaksiRepository;
22.
23.         $this->middleware('auth');
24.         $this->middleware('petani');
25.
26.     }
27.
28.     public function showAll($date){
29.         $transaksi=$this->transaksiRepository-
    >hargaBarangByPetani($date);
30.         $tanggal=$this->transaksiRepository-
    >getTglByIdPetani();
31.         $total=0;
32.
33.         foreach($transaksi as $tgl){
34.             $total+=$tgl->harga;
35.         }
36.
37.         $data=[
38.             'filter_tanggal' =>$date,
39.             'transaksi' => $transaksi,
40.             'tanggal' => $tanggal,
41.             'total'=>$total
42.         ];
43.
44.         // dd($data);
45.
```

```
46.          return view('petani.pembayaran')-
    >with('data',$data);
47.      }
48.
49. }
```

Kode Sumber 5.60 PembayaranController

5.3.4. Role Reseller

merupakan kumpulan *Controller* yang dikhususkan untuk role Reseller. Berikut adalah beberapa *Controller*:

5.3.4.1. AlamatController

Lapisan ini bertugas untuk mengatur Alamat Reseller. Kode Sumber 5.61 berikut merupakan implementasi dari lapisan kontrol AlamatController:

```
1.    <?php
2.
3.    namespace App\Http\Controllers\Reseller;
4.
5.    use App\Http\Controllers\Controller;
6.    use Illuminate\Http\Request;
7.    use App\Repositories\AlamatRepository;
8.    use Illuminate\Support\Facades\DB;
9.
10.   use App\Repositories\KategoriRepository;
11.   use App\Repositories\ProdukRepository;
12.   use App\Repositories\BaseKategoriRepository;
13.   use Illuminate\Support\Facades\Auth;
14.
15.   class AlamatController extends Controller
16.   {
17.       private $alamat;
```

```
18.
19.         public function __construct(AlamatRepository $a
        lamat, KategoriRepository $kategoriRepository, Prod
        ukRepository $produkRepository, BaseKategoriReposit
        ory $baseKategoriRepository)
20.         {
21.             $this-
        >kategoriRepository = $kategoriRepository;
22.             $this-
        >produkRepository = $produkRepository;
23.             $this-
        >baseKategoriRepository = $baseKategoriRepository;

24.             $this->middleware('auth');
25.             $this->middleware('reseller');
26.
27.             $this->alamat = $alamat;
28.         }
29.
30.         public function showAlamat()
31.         {
32.             $data = [
33.                 'alamat' => $this->alamat-
        >getAllAlamatByUser(Auth::user()->id)
34.             ];
35.
36.             return view('reseller.alamat')-
        >with(compact('data'));
37.         }
38.
39.         public function showAlamatbyUser($id)
40.         {
41.             $alamat = [
42.                 'alamat' => $this->alamat-
        >getAllAlamatByUser($id)
43.             ];
44.
```

```
45.             $kategori = [
46.                 'sayur' => $this->kategoriRepository->kategoriSayur(),
47.                 'buah' => $this->kategoriRepository->kategoriBuah(),
48.                 'makanansehat' => $this->kategoriRepository->kategoriMakananSehat(),
49.                 'minumansehat' => $this->kategoriRepository->kategoriMinumanSehat(),
50.                 'beras' => $this->kategoriRepository->kategoriBeras(),
51.                 'bahanolahan' => $this->kategoriRepository->kategoriBahanOlahan(),
52.                 'berkebun' => $this->kategoriRepository->kategoriBerkebun(),
53.                 'lainlain' => $this->kategoriRepository->kategoriLainLain()
54.             ];
55.
56.             $data = [
57.                 'barang' => $this->produkRepository->showEtalase(),
58.                 'barang_paket' => $this->produkRepository->showByJenis('Paket'),
59.                 'barang_timbang' => $this->produkRepository->showByJenis('Timbang'),
60.                 'barang_kemas' => $this->produkRepository->showByJenis('Kemas'),
61.                 'kategori' => $this->kategoriRepository->array(),
62.                 'baseKategori' => $this->baseKategoriRepository->all(),
63.             ];
64.
65.             //dd($alamat);
66.
```

```
67.             return view('reseller.alamat')-
    >with(compact('alamat'))
68.                 ->with(compact('data'))
69.                 ->with(compact('kategori'));
70.         }
71.
72.         public function hapusAlamat($id)
73.         {
74.             $this->alamat->hapusAlamat($id);
75.
76.             return redirect('Reseller/LihatAlamat/'.Aut
    h::user()->id);
77.         }
78.
79.         public function _tambahAlamat(Request $request)

80.         {
81.             //dd($request);
82.             try
83.             {
84.                 DB::beginTransaction();
85.
86.                 $this->alamat->tambahAlamat($request-
    >all());
87.
88.                 DB::commit();
89.
90.                 return redirect('Reseller/LihatAlamat/'
    .Auth::user()->id);
91.             }
92.             catch (\Throwable $th)
93.             {
94.                 DB::rollback();
95.
96.                 return $th;
97.             }
98.         }
```

```
99.
100.        public function tambahAlamat()
101.        {
102.            return view('reseller.tambah-alamat');
103.        }
104.
105.        public function ubahAlamat($id)
106.        {
107.            $alamat=$this->alamat-
        >getAlamatById($id);
108.            return view('reseller.ubah-alamat')-
        >with(compact('alamat'));
109.        }
110.
111.        public function _ubahAlamat($id, Request $reque
        st)
112.        {
113.            try
114.            {
115.                DB::beginTransaction();
116.
117.                $this->alamat-
        >ubahAlamat($id, $request->all());
118.
119.                DB::commit();
120.
121.                return redirect('Reseller/LihatAlamat/'
        .Auth::user()->id);
122.            }
123.            catch (\Throwable $th)
124.            {
125.                DB::rollback();
126.
127.                return $th;
128.            }
129.        }
130.    }
```

Kode Sumber 5.61 AlamatController

5.3.4.2. EtalaseController

Lapisan ini bertugas untuk mengatur Etalase Reseller. Kode Sumber 5.62 berikut merupakan implementasi dari lapisan kontrol EtalaseController:

```
<?php

namespace App\Http\Controllers\Reseller;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Repositories\ProdukRepository;
use App\Repositories\KategoriRepository;
use App\Repositories\BaseKategoriRepository;
use App\Repositories\EtalaseRepository;
use App\Repositories\KeranjangRepository;
use App\Repositories\BarangKemasanRepository;
use App\Repositories\IsiPaketRepository;
use App\Repositories\TanggalRepository;
use App\Repositories\AlamatRepository;
use Carbon\Carbon;

use Illuminate\Support\Facades\Auth;
use DB;

class EtalaseController extends Controller
{
    protected $produkRepository,
              $kategoriRepository,
              $baseKategoriRepository,
              $etalaseRepository,
              $barangKemasanRepository,
              $isiPaketRepository,
              $keranjangRepository;
```

```
30.
31.         public function __construct(BarangKemasanReposi
    tory $barangKemasanRepository, ProdukRepository $pr
    odukRepository, KategoriRepository $kategoriReposit
    ory, BaseKategoriRepository $baseKategoriRepository
    , EtalaseRepository $etalaseRepository, KeranjangRe
    pository $keranjangRepository, IsiPaketRepository $
    isiPaketRepository, TanggalRepository $tanggal, Ala
    matRepository $alamat)
32.         {
33.             $this-
    >produkRepository = $produkRepository;
34.             $this-
    >kategoriRepository = $kategoriRepository;
35.             $this-
    >baseKategoriRepository = $baseKategoriRepository;

36.             $this-
    >etalaseRepository = $etalaseRepository;
37.             $this-
    >keranjangRepository = $keranjangRepository;
38.             $this-
    >barangKemasanRepository = $barangKemasanRepository
    ;
39.             $this-
    >isiPaketRepository = $isiPaketRepository;
40.             $this->tanggal = $tanggal;
41.             $this->alamat = $alamat;
42.
43.             $this->middleware('auth');
44.         }
45.
46.         public function etalase()
47.         {
48.             return redirect('/Reseller/Etalase/'.date('
    Y-m-d'));
49.         }
```

```
50.
51.         public function etalaseDate($date)
52.         {
53.             $tanggal= Carbon::now()->format('yy-m-
    d');
54.             $hapus=$this->tanggal-
    >hapusTanggal($tanggal);
55.             $getAlamat = $this->alamat-
    >getAllAlamatByUser(Auth::user()->id);
56.             $data = [
57.                 'barang' => DB::select('call sp_get_bar
    ang_by_tanggal(?)',array($date)),
58.                 'barang_paket' => DB::select('call sp_g
    et_barang_by_tanggal_and_jenis(?, ?)',array($date,
    'Paket')),
59.                 'barang_timbang' => DB::select('call sp
    _get_barang_by_tanggal_and_jenis(?, ?)',array($date
    , 'Timbang')),
60.                 'barang_kemas' => DB::select('call sp_g
    et_barang_by_tanggal_and_jenis(?, ?)',array($date,
    'Kemas')),
61.                 'kategori' => $this-
    >kategoriRepository->array(),
62.                 'baseKategori' => $this-
    >baseKategoriRepository->all(),
63.                 'tanggal_pengiriman' => $this-
    >getTanggalPengiriman(),
64.                 'alamat'                => $getAlamat,

65.                 'tanggal'=>$date,
66.             ];
67.
68.             return view('reseller.etalase')
69.                 ->with(compact('data'));
70.         }
71.
72.         public function getTanggalPengiriman()
```

```
73.         {
74.
75.             $getListHari=$this->tanggal-
    >get_tanggal();
76.             // dd($getListHari);
77.             if(count($getListHari) > 0){
78.                 for($a=0;$a<sizeof($getListHari);$a++)

79.                 {
80.                     $tgl=$getListHari[$a]['tanggal'];
81.                     // string
82.                     $finalListHari[$a]['tanggal'] = Car
    bon::parse($tgl)->format('D, d F Y');
83.                     //tanggal value
84.                     $finalListHari[$a]['tanggal_value']
    = Carbon::parse($tgl)->format('Y-m-d');
85.
86.                     // dd($finalListHari[$a]['tanggal_v
    alue']);
87.                 }
88.             }
89.             else{
90.                 $finalListHari[0]['tanggal'] = Carbon::
    now()->format('D, d F Y');
91.                     //tanggal value
92.                 $finalListHari[0]['tanggal_value'] = Ca
    rbon::now()->format('Y-m-d');
93.             }
94.
95.
96.
97.
98.
99.             foreach ($finalListHari as $key => $value)

100.            {
```

```
101.                 $finalListHari[$key] = (object) $value;

102.             }
103.
104.             // dd($finalListHari);
105.
106.             return $finalListHari;
107.         }
108.
109.         public function cariEtalase($nama)
110.         {
111.             return redirect('/Reseller/Etalase/CariProduk/'.$nama.'/'.date('Y-m-d'));
112.             $data = [
113.                 'pencarian_produk' => $nama,
114.                 'barang' => $this->produkRepository->findByNamaBarang($nama),
115.                 'kategori' => $this->kategoriRepository->array(),
116.                 'baseKategori' => $this->baseKategoriRepository->all()
117.             ];
118.
119.             return view('reseller.cariproduk')
120.                 ->with(compact('data'));
121.         }
122.
123.         public function cariEtalaseDate($nama, $date)
124.         {
125.             $data = [
126.                 'pencarian_produk' => $nama,
127.                 'barang' => DB::select('call sp_get_barang_by_tanggal_and_search(?, ?)',array($date, $nama)),
128.                 'kategori' => $this->kategoriRepository->array(),
```

```
129.                 'baseKategori' => $this->baseKategoriRepository->all(),
130.                 'tanggal_pengiriman' => $this->getTanggalPengiriman(),
131.                 'tanggal'=>$date
132.             ];
133.
134.             return view('reseller.cariproduk')
135.                 ->with(compact('data'));
136.         }
137.
138.         public function detailProduk($id_barang)
139.         {
140.             $get_barang = $this->produkRepository->findById($id_barang);
141.             // dd($get_barang);
142.
143.             if($get_barang->jenis == 'Timbang')
144.             {
145.                 $barang_kemasan = $this->barangKemasanRepository->findByIdBarang($id_barang);
146.
147.                 $data = [
148.                     'barang' => $get_barang,
149.                     'kategori' => $this->kategoriRepository->array(),
150.                     'baseKategori' => $this->baseKategoriRepository->all()
151.                 ];
152.
153.                 return view('reseller.detail-produk-timbang')
154.                     ->with(compact('data'));
155.             }
156.
157.             else if($get_barang->jenis == 'Kemas')
```

```
158.          {
159.              $barang_kemasan = $this->barangKemasanRepository->findByIdBarang($id_barang);
160.
161.              $data = [
162.                  'barang' => $get_barang,
163.                  'kemasan' => $barang_kemasan,
164.                  'kategori' => $this->kategoriRepository->array(),
165.                  'baseKategori' => $this->baseKategoriRepository->all()
166.              ];
167.
168.              return view('reseller.detail-produk-kemas')
169.                  ->with(compact('data'));
170.          }
171.
172.          else if($get_barang->jenis == 'Paket')
173.          {
174.              $data = [
175.                  'barang' => $get_barang,
176.                  'isi_paket' => $this->isiPaketRepository->read($get_barang->id),
177.                  'kategori' => $this->kategoriRepository->array(),
178.                  'baseKategori' => $this->baseKategoriRepository->all()
179.              ];
180.
181.              return view('reseller.detail-produk-paket')
182.                  ->with(compact('data'));
183.          }
184.      }
185.  }
```

Kode Sumber 5.62 EtalaseController

5.3.4.3. KeranjangController

Lapisan ini bertugas untuk mengatur Keranjang Reseller. Kode Sumber 5.63 berikut merupakan implementasi dari lapisan kontrol KeranjangController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Reseller;
4.
5.  use Carbon\Carbon;
6.  use App\Http\Controllers\Controller;
7.  use Illuminate\Http\Request;
8.  use Illuminate\Support\Facades\DB;
9.  use Webpatser\Uuid\Uuid;
10. use Illuminate\Support\Facades\Auth;
11. use App\Repositories\BarangKemasanRepository;
12. use App\Repositories\KeranjangRepository;
13. use App\Repositories\DetailKeranjangRepository;
14. use App\Repositories\BobotKemasanRepository;
15. use App\Repositories\ResepRepository;
16. use App\Repositories\IsiResepRepository;
17. use App\Repositories\ProdukRepository;
18. use App\Repositories\IsiPaketRepository;
19. use App\Repositories\HariPengirimanRepository;
20. use App\Repositories\TanggalRepository;
21. use App\Repositories\KeranjangResellerRepository;
22. use App\Repositories\ParentKeranjangResellerReposit
    ory;
23.
24.
25. use App\Models\Detail_keranjang as DetailKeranjang;

26.
27.
```

```
28.    class KeranjangController extends Controller
29.    {
30.        protected $keranjang;
31.        protected $detailKeranjang;
32.        protected $bobotKemasan;
33.        protected $resep;
34.        protected $isiResep;
35.        protected $barang;
36.        protected $isiPaket;
37.        protected $hariPengiriman;
38.        protected $barangKemasan;
39.        protected $tanggal;
40.        protected $keranjangReseller;
41.        protected $parentKeranjangReseller;
42.
43.        public function __construct(BarangKemasanReposi
       tory $barangKemasan, HariPengirimanRepository $hari
       Pengiriman ,KeranjangRepository $keranjang, DetailK
       eranjangRepository $detailKeranjang, BobotKemasanRe
       pository $bobotKemasan, ResepRepository $resep, Isi
       ResepRepository $isiResep, ProdukRepository $barang
       , IsiPaketRepository $isiPaket, TanggalRepository $
       tanggal, KeranjangResellerRepository $keranjangRese
       ller, ParentKeranjangResellerRepository $parentKera
       njangReseller)
44.        {
45.            $this->middleware('auth');
46.
47.            $this->keranjang         = $keranjang;
48.            $this-
       >detailKeranjang    = $detailKeranjang;
49.            $this->bobotKemasan      = $bobotKemasan;
50.            $this->resep             = $resep;
51.            $this->isiResep          = $isiResep;
52.            $this->barang            = $barang;
53.            $this->isiPaket          = $isiPaket;
```

```
54.             $this->hariPengiriman    = $hariPengiriman;
55.             $this->barangKemasan     = $barangKemasan;
56.             $this->tanggal           = $tanggal;
57.             $this->keranjangReseller = $keranjangReseller;
58.             $this->parentKeranjangReseller = $parentKeranjangReseller;
59.         }
60.
61.         public function tambahItemKeranjang($idKeranjang,Request $request)
62.         {
63.             // dd($request->all());
64.             try
65.             {
66.                 DB::beginTransaction();
67.
68.                 $this->detailKeranjang->tambahDetailKeranjang($request->all());
69.
70.                 DB::commit();
71.
72.                 return 'ok';
73.             }
74.             catch (\Throwable $th)
75.             {
76.                 DB::rollback();
77.
78.                 return 'failed';
79.             }
80.         }
81.
82.         public function ubahItemKeranjang($id, Request $request)
```

```
83.         {
84.             try
85.             {
86.                 DB::beginTransaction();
87.
88.                 $this->detailKeranjang-
        >updateDetailKeranjang($id, $request->all());
89.
90.                 DB::commit();
91.
92.                 return 1;
93.             }
94.             catch (\Throwable $th)
95.             {
96.                 DB::rollback();
97.
98.                 return 0;
99.             }
100.        }
101.
102.        public function hapusItemKeranjang($id)
103.        {
104.            try
105.            {
106.                DB::beginTransaction();
107.
108.                $this->detailKeranjang-
        >hapusDetailKeranjang($id);
109.
110.                DB::commit();
111.
112.                return 'ok';
113.            }
114.            catch (\Throwable $th)
115.            {
116.                DB::rollback();
117.
```

```
118.                 return 'failed';
119.             }
120.         }
121.
122.         public function getBobotKemasan()
123.         {
124.             return $this->bobotKemasan->all();
125.         }
126.
127.         public function showInputItemKeranjang($id)
128.         {
129.             $cekBarang          = $this->barang-
    >findById($id);
130.
131.             if($cekBarang->jenis == 'Kemas')
132.             {
133.
134.                 $data = [
135.                     'barang' => $cekBarang,
136.                     // 'bobot_kemasan' => $this-
    >bobotKemasan->all()
137.                     'bobot_kemasan' => $this-
    >barangKemasan->findByIdBarang($id)
138.                 ];
139.
140.                 return view('reseller.components.input-
    item-kemas')
141.                     ->with(compact('data'))
142.                     ->render();
143.             }
144.             else if($cekBarang->jenis == 'Paket')
145.             {
146.                 $getIsiPaket = $this->isiPaket-
    >findByIdBarang($id);
147.
148.                 foreach($getIsiPaket as $isiPaket)
149.                 {
```

```
150.                 $getBarang             = $this->barang->findById($isiPaket->id_barang);
151.
152.                 $isiPaket->nama_barang = $getBarang->nama;
153.                 $isiPaket->satuan      = $getBarang->satuan;
154.             }
155.
156.             $data = [
157.                 'barang'    => $cekBarang,
158.                 'isiPaket'  => $getIsiPaket
159.             ];
160.
161.             return view('reseller.components.input-item-paket')
162.                 ->with(compact('data'))
163.                 ->render();
164.         }
165.         else if($cekBarang->jenis == 'Timbang')
166.         {
167.             $data = [
168.                 'barang' => $cekBarang,
169.             ];
170.
171.             return view('reseller.components.input-item-timbang')
172.                 ->with(compact('data'))
173.                 ->render();
174.         }
175.     }
176.
177.     public function showUbahItemKeranjang($id)
178.     {
179.
180.         $getDetailKeranjang = $this->detailKeranjang->findById($id);
```

```
181.          $cekBarang = $this->barang->findById($getDetailKeranjang->id_barang);
182.
183.          if($cekBarang->jenis == 'Kemas')
184.          {
185.
186.              $data = [
187.                  'barang' => $cekBarang,
188.                  'bobot_kemasan' => $this->barangKemasan->findByIdBarang($getDetailKeranjang->id_barang),
189.                  'isi_keranjang' => $getDetailKeranjang
190.              ];
191.
192.              return view('reseller.components.ubah-item-kemas')
193.                  ->with(compact('data'))
194.                  ->render();
195.          }
196.          else if($cekBarang->jenis == 'Paket')
197.          {
198.              $getIsiPaket = $this->isiPaket->findByIdBarang($getDetailKeranjang->id_barang);
199.
200.              foreach($getIsiPaket as $isiPaket)
201.              {
202.                  $getBarang             = $this->barang->findById($isiPaket->id_barang);
203.
204.                  $isiPaket->nama_barang = $getBarang->nama;
205.                  $isiPaket->satuan      = $getBarang->satuan;
206.              }
207.
208.              $data = [
```

```
                    'barang'    => $cekBarang,
                    'isiPaket'  => $getIsiPaket,
                    'isi_keranjang' => $getDetailKeranjang
                ];

                return view('reseller.components.ubah-item-paket')
                    ->with(compact('data'))
                    ->render();
            }
            else if($cekBarang->jenis == 'Timbang')
            {
                $data = [
                    'barang' => $cekBarang,
                    'isi_keranjang' => $getDetailKeranjang
                ];

                return view('reseller.components.ubah-item-timbang')
                    ->with(compact('data'))
                    ->render();
            }
        }

        public function submitInputItemKeranjang($id, Request $request)
        {
            // dd($this->detailKeranjang->findByIdBarangBobotKemasan($id, (int) $request->volume_order));

            // if($this->barang->findById($id)->diskon>100){
```

```
236.          //     $harga_diskon=$this->barang->findById($id)->harga_jual * $request->total_order;
237.          // }
238.
239.
240.          try
241.          {
242.              DB::beginTransaction();
243.
244.              if($this->keranjang->findKeranjang($request->tanggal) == null)
245.              {
246.                  $this->keranjang->tambahKeranjang($request->tanggal);
247.              }
248.
249.              $getIdKeranjang = $this->keranjang->getIDKeranjang($request->tanggal);
250.
251.              switch($this->barang->findById($id)->jenis)
252.              {
253.                  case 'Paket':
254.                      if($this->barang->findById($id)->diskon>100){
255.                          $harga_diskon=($this->barang->findById($id)->harga_jual-$this->barang->findById($id)->diskon) * $request->total_order;
256.                      }
257.                      else{
258.                          $harga_diskon=($this->barang->findById($id)->harga_jual-($this->barang->findById($id)->harga_jual*($this->barang->findById($id)->diskon/100))) * $request->total_order;
259.                      }
```

```
260.                     if($this->detailKeranjang->findByIdKeranjangIdBarang($getIdKeranjang,$id) == null)
261.                     {
262.                         $data = [
263.                             'id'            => Uuid::generate(),
264.                             'id_keranjang'  => $getIdKeranjang,
265.                             'id_barang'     => $id,
266.                             'volume'        => $request->total_order,
267.                             'harga'         => $this->barang->findById($id)->harga_jual * $request->total_order,
268.                             'harga_diskon'  => $harga_diskon
269.                         ];
270.
271.                         $this->detailKeranjang->tambahDetailKeranjang($data);
272.
273.                         DB::commit();
274.                         return 1;
275.                     }
276.                     else
277.                     {
278.                         $currentVolume = $this->detailKeranjang->findByIdKeranjangIdBarang($getIdKeranjang,$id)->volume;
279.                         $currentHarga  = $this->detailKeranjang->findByIdKeranjangIdBarang($getIdKeranjang,$id)->harga;
```

```
280.                            $currentHargaDiskon  = $this->detailKeranjang->findByIdKeranjangIdBarang($getIdKeranjang,$id)->harga_diskon;
281.
282.                            $data = [
283.                                'volume'        => $request->total_order + $currentVolume,
284.                                'harga'         => ($this->barang->findById($id)->harga_jual * $request->total_order) + $currentHarga,
285.                                'harga_diskon' => $harga_diskon+$currentHargaDiskon,
286.                            ];
287.
288.                            $this->detailKeranjang->updateDetailKeranjang($id, $data);
289.
290.                            DB::commit();
291.                            return 1;
292.                        }
293.
294.                    case 'Timbang':
295.                        if($this->barang->findById($id)->diskon>100){
296.                            $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-($this->barang->findById($id)->diskon/10)) * ($request->total_order/100);
297.                        }
298.                        else{
299.                            $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-(($this->barang->findById($id)->harga_jual/10)*($this->barang->findById($id)->diskon/100))) * (($request->total_order)/100);
300.                        }
```

```
301.
302.                     if($this->detailKeranjang->findByIdKeranjangIdBarang($getIdKeranjang,$id) == null)
303.                     {
304.                         $data = [
305.                             'id'            => Uuid::generate(),
306.                             'id_keranjang' => $getIdKeranjang,
307.                             'id_barang'     => $id,
308.                             'volume'        => $request->total_order,
309.                             'harga'         => ($this->barang->findById($id)->harga_jual/10) * (($request->total_order)/100),
310.                             'harga_diskon' => $harga_diskon
311.                         ];
312.
313.                         $this->detailKeranjang->tambahDetailKeranjang($data);
314.
315.                         DB::commit();
316.                         return 1;
317.                     }
318.                     else
319.                     {
320.                         $currentVolume = $this->detailKeranjang->findByIdKeranjangIdBarang($getIdKeranjang,$id)->volume;
321.                         $currentHarga  = $this->detailKeranjang->findByIdKeranjangIdBarang($getIdKeranjang,$id)->harga;
```

```
322.                             $currentHargaDiskon  = $this->detailKeranjang->findByIdKeranjangIdBarang($getIdKeranjang,$id)->harga_diskon;
323.
324.                             $data = [
325.                                 'volume'        => $request->total_order + $currentVolume,
326.                                 'harga'         => ($this->barang->findById($id)->harga_jual/10) * (($request->total_order)/100) + $currentHarga,
327.                                 'harga_diskon' => $harga_diskon + $currentHargaDiskon
328.                             ];
329.
330.                             $this->detailKeranjang->updateDetailKeranjang($id, $data);
331.
332.                             DB::commit();
333.                             return 1;
334.                         }
335.
336.                     case 'Kemas':
337.                         if($this->barang->findById($id)->diskon>100){
338.                             $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-($this->barang->findById($id)->diskon/10)) * (($request->total_order*$request->volume_order)/100);
339.                         }
340.                         else{
341.                             $harga_diskon=(($this->barang->findById($id)->harga_jual/10)-(($this->barang->findById($id)->harga_jual/10)*($this->barang->findById($id)->diskon/100))) * (($request->total_order*$request->volume_order)/100);
```

```
342.                     }
343.                     if($this->detailKeranjang->findByIdBarangBobotKemasan($getIdKeranjang, $id, (int) $request->volume_order) == null)
344.                     {
345.                         $data = [
346.                             'id'            => Uuid::generate(),
347.                             'id_keranjang' => $getIdKeranjang,
348.                             'id_barang'     => $id,
349.                             'volume'        => $request->total_order,
350.                             'bobot_kemasan'=> $request->volume_order,
351.                             'harga'         => ($this->barang->findById($id)->harga_jual/10) * (($request->total_order*$request->volume_order)/100),
352.                             'harga_diskon' => $harga_diskon
353.                         ];
354.
355.                         $this->detailKeranjang->tambahDetailKeranjang($data);
356.
357.                         DB::commit();
358.                         return 1;
359.                     }
360.                     else
361.                     {
362.                         $currentVolume = $this->detailKeranjang->findByIdBarangBobotKemasan($getIdKeranjang,$id,(int) $request->volume_order)->volume;
```

```
363.                            $currentHarga  = $this->detailKeranjang->findByIdBarangBobotKemasan($getIdKeranjang,$id,(int) $request->volume_order)->harga;
364.                            $currentHargaDiskon  = $this->detailKeranjang->findByIdBarangBobotKemasan($getIdKeranjang,$id,(int) $request->volume_order)->harga_diskon;
365.
366.                            $data = [
367.                                'volume'       => $request->total_order + $currentVolume,
368.                                'harga'        => (($this->barang->findById($id)->harga_jual/10) * (($request->total_order*$request->volume_order)/100)) + $currentHarga,
369.                                'harga_diskon' => $harga_diskon+$currentHargaDiskon
370.                            ];
371.
372.                            $this->detailKeranjang->updateDetailKeranjangKemas($id,(int) $request->volume_order,$data);
373.
374.                            DB::commit();
375.                            return 1;
376.                        }
377.                    }
378.            }
379.            catch (\Throwable $th)
380.            {
381.                DB::rollback();
382.                return $th->getMessage();
383.            }
384.        }
385.
```

```
386.      public function submitUbahItemKeranjang($id, Re
      quest $request)
387.      {
388.          try
389.          {
390.              DB::beginTransaction();
391.
392.              if($this->keranjang-
      >findKeranjang($request->tanggal) == null)
393.              {
394.                  $this->keranjang-
      >tambahKeranjang($request->tanggal);
395.              }
396.
397.              $getIdKeranjang = $this->keranjang-
      >getIDKeranjang($request->tanggal);
398.
399.              switch($this->barang->findById($id)-
      >jenis)
400.              {
401.                  case 'Paket':
402.                          if($this->barang-
      >findById($id)->diskon>100){
403.                              $harga_diskon=($this-
      >barang->findById($id)->harga_jual-$this->barang-
      >findById($id)->diskon) * $request->total_order;
404.                          }
405.                          else{
406.                              $harga_diskon=($this-
      >barang->findById($id)->harga_jual-($this->barang-
      >findById($id)->harga_jual*($this->barang-
      >findById($id)->diskon/100))) * $request-
      >total_order;
407.                          }
408.                          $data = [
409.                              'id'            => Uuid:
      :generate(),
```

```
                                'id_keranjang' => $getIdKeranjang,
                                'id_barang'    => $id,
                                'volume'       => $request->total_order,
                                'harga'        => $this->barang->findById($id)->harga_jual * $request->total_order,
                                'harga_diskon' => $harga_diskon,
                            ];

                            $this->detailKeranjang->updateDetailKeranjang($id, $data);

                            DB::commit();
                            return 1;

                    case 'Timbang':
                            $data = [
                                'id'           => Uuid::generate(),
                                'id_keranjang' => $getIdKeranjang,
                                'id_barang'    => $id,
                                'volume'       => $request->total_order,
                                'harga'        => ($this->barang->findById($id)->harga_jual/10) * (($request->total_order)/100),
                                'harga_diskon' => (($this->barang->findById($id)->harga_jual/10)-(($this->barang->findById($id)->harga_jual/10)*($this->barang->findById($id)->diskon/100))) * (($request->total_order)/100)
```

```
430.                             ];
431.
432.                             $this->detailKeranjang-
    >updateDetailKeranjang($id, $data);
433.
434.                             DB::commit();
435.                             return 1;
436.
437.                     case 'Kemas':
438.                             $data = [
439.                                 'id'            => Uuid:
    :generate(),
440.                                 'id_keranjang' => $getI
    dKeranjang,
441.                                 'id_barang'     => $id,

442.                                 'volume'        => $requ
    est->total_order,
443.                                 'bobot_kemasan'=> $requ
    est->volume_order,
444.                                 'harga'         => ($thi
    s->barang->findById($id)-
    >harga_jual/10) * (($request->total_order*$request-
    >volume_order)/100),
445.                                 'harga_diskon' => (($th
    is->barang->findById($id)->harga_jual/10)-(($this-
    >barang->findById($id)->harga_jual/10)*($this-
    >barang->findById($id)->diskon/100))) * (($request-
    >total_order*$request->volume_order)/100)
446.                             ];
447.
448.                             $this->detailKeranjang-
    >updateDetailKeranjang($id, $data);
449.
450.                             DB::commit();
451.                             return 1;
452.                     }
```

```
453.            }
454.            catch (\Throwable $th)
455.            {
456.                DB::rollback();
457.                return $th->getMessage();
458.            }
459.        }
460.
461.        public function lihatItemKeranjang($date)
462.        {
463.            if($this->keranjang-
    >findKeranjang($date) == null){
464.                $this->keranjang-
    >tambahKeranjang($date);
465.            }
466.
467.            $getIDKeranjang       = $this->keranjang-
    >getIDKeranjang($date);
468.            $getDetailKeranjang   = $this-
    >detailKeranjang-
    >getDetailKeranjang($getIDKeranjang);
469.            $total=0;
470.
471.            foreach($getDetailKeranjang as $detailKeran
    jang)
472.            {
473.                $getBarang = $this->barang-
    >findById($detailKeranjang->id_barang);
474.                if($getBarang->jenis == 'Kemas')
475.                {
476.                    $detailKeranjang-
    >nama       = $getBarang-
    >nama.' ('.$detailKeranjang-
    >bobot_kemasan.' '.'Gram'.')';
477.                    $detailKeranjang-
    >jenis      = $getBarang->jenis;
```

```
478.                     $detailKeranjang->satuan     = 'Kemas';
479.                     $detailKeranjang->bobot      = $getBarang->bobot;
480.                     $detailKeranjang->volume     = $detailKeranjang->volume;
481.                 }
482.                 else if($getBarang->jenis == 'Paket' || $getBarang->jenis == 'Timbang')
483.                 {
484.                     $detailKeranjang->nama       = $getBarang->nama;
485.                     $detailKeranjang->jenis      = $getBarang->jenis;
486.                     $detailKeranjang->satuan     = $getBarang->satuan;
487.                     $detailKeranjang->bobot      = $getBarang->bobot;
488.                 }
489.                 $total+= $detailKeranjang->harga_diskon;
490.             }
491.
492.             $data = [
493.                 'total'=>$total,
494.                 'detail_keranjang'  => $getDetailKeranjang,
495.                 'tanggal_pengiriman' => $this->getTanggalPengiriman(),
496.                 'tanggal_pengirimans' => $date
497.             ];
498.
499.             return view('reseller.components.lihat-keranjang')
500.                 ->with(compact('data'))
501.                 ->render();
```

```
502.        }
503.
504.        public function getTanggalPengiriman()
505.        {
506.
507.            $getListHari=$this->tanggal-
    >get_tanggal();
508.
509.            for($a=0;$a<sizeof($getListHari);$a++)
510.            {
511.                $tgl=$getListHari[$a]['tanggal'];
512.                // string
513.                $finalListHari[$a]['tanggal'] = Carbon:
    :parse($tgl)->format('D, d F Y');
514.                //tanggal value
515.                $finalListHari[$a]['tanggal_value'] = C
    arbon::parse($tgl)->format('Y-m-d');
516.
517.                // dd($finalListHari[$a]['tanggal_value
    ']);
518.            }
519.
520.
521.
522.            foreach ($finalListHari as $key => $value)

523.            {
524.                $finalListHari[$key] = (object) $value;

525.            }
526.
527.            // dd($finalListHari);
528.
529.            return $finalListHari;
530.        }
531.
532.        public function simpan(Request $request){
```

```
533.            // dd($request->tanggal);
534.            $getKeranjang           = $this->keranjang-
    >getIDKeranjang($request->tanggal);
535.            $getDetailKeranjang     = $this-
    >detailKeranjang-
    >getDetailKeranjang($getKeranjang);
536.            // $getKeranjangReseller   = $this-
    >keranjangReseller->get(Auth::user()->id)
537.
538.            $datas=[
539.                'name'=>$request->name,
540.                'alamat' =>$request->alamat,
541.                'nohp' =>$request->nohp,
542.                'tanggal_pre_order' => $request-
    >tanggal,
543.            ];
544.            //cek nama
545.            $cek_nama=$this->parentKeranjangReseller-
    >getIdFromName($request->name, $request-
    >tanggal);
546.            // dd($cek_nama);
547.            if($cek_nama==null){
548.                $this->parentKeranjangReseller-
    >create($datas);
549.            }
550.
551.            $getParentKeranjang = $this-
    >parentKeranjangReseller->getIDKeranjang($request-
    >name, $request->tanggal);
552.
553.            foreach($getDetailKeranjang as $detail){
554.                // dd($detail->id_barang);
555.                // if($detail->id_barang==)
556.
557.                //cek barang
```

```
558.                $cek_barang=$this->keranjangReseller->getBarangFromIdBarangAndBobot($getParentKeranjang, $detail->id_barang, $detail->bobot_kemasan);
559.                // dd($cek_barang);
560.                if($cek_barang==null){
561.                    $data = [
562.                        'id_parent_keranjang'    => $getParentKeranjang,
563.                        'id_barang' => $detail->id_barang,
564.                        'volume' =>$detail->volume,
565.                        'harga' =>$detail->harga,
566.                        'bobot_kemasan' => $detail->bobot_kemasan,
567.                        'harga_diskon' =>$detail->harga_diskon,
568.                    ];
569.                    $this->keranjangReseller->create($data);
570.                }
571.                else{
572.                    $data=[
573.                        'volume' =>$detail->volume + $cek_barang->volume,
574.                        'harga' =>$detail->harga + $cek_barang->harga,
575.                        'harga_diskon' =>$detail->harga_diskon + $cek_barang->harga_diskon,
576.                    ];
577.                    $this->keranjangReseller->update($cek_barang->id,$data);
578.                }
579.
580.                // DB::commit();
581.            }
582.            $this->keranjang->hapusKeranjang($getKeranjang);
```

```
583.            $this->detailKeranjang-
    >hapusDetailKeranjangByIdKeranjang($getKeranjang);

584.            return redirect()->back();
585.            // }
586.            // catch (\Throwable $th)
587.            // {
588.            //     DB::rollback();
589.            //     return $th->getMessage();
590.            // }
591.            // dd($getDetailKeranjang);
592.        }
593.    }
```

Kode Sumber 5.63 KeranjangController

5.3.4.4. ProfilController

Lapisan ini bertugas untuk mengatur Profil Reseller. Kode Sumber 5.64 berikut merupakan implementasi dari lapisan kontrol ProfilController:

```
1.    <?php
2.
3.    namespace App\Http\Controllers\Reseller;
4.
5.    use App\Http\Controllers\Controller;
6.    use Illuminate\Http\Request;
7.    use Auth;
8.    use DB;
9.
10.   use App\Repositories\KategoriRepository;
11.   use App\Repositories\ProdukRepository;
12.   use App\Repositories\BaseKategoriRepository;
13.   use App\Repositories\UserRepository;
14.
15.   class ProfilController extends Controller
```

```
16.    {
17.        protected $kategoriRepository, $produkRepositor
       y, $baseKategoriRepository, $userRepository;
18.
19.        public function __construct(KategoriRepository
       $kategoriRepository, ProdukRepository $produkReposi
       tory, BaseKategoriRepository $baseKategoriRepositor
       y, UserRepository $userRepository)
20.        {
21.            $this-
       >kategoriRepository = $kategoriRepository;
22.            $this-
       >produkRepository = $produkRepository;
23.            $this-
       >baseKategoriRepository = $baseKategoriRepository;

24.            $this->userRepository=$userRepository;
25.
26.            $this->middleware('auth');
27.            $this->middleware('reseller');
28.        }
29.
30.        public function viewProfile()
31.        {
32.            $kategori = [
33.                'sayur' => $this->kategoriRepository-
       >kategoriSayur(),
34.                'buah' => $this->kategoriRepository-
       >kategoriBuah(),
35.                'makanansehat' => $this-
       >kategoriRepository->kategoriMakananSehat(),
36.                'minumansehat' => $this-
       >kategoriRepository->kategoriMinumanSehat(),
37.                'beras' => $this->kategoriRepository-
       >kategoriBeras(),
38.                'bahanolahan' => $this-
       >kategoriRepository->kategoriBahanOlahan(),
```

```
39.                 'berkebun' => $this->kategoriRepository->kategoriBerkebun(),
40.                 'lainlain' => $this->kategoriRepository->kategoriLainLain()
41.             ];
42.
43.             $data = [
44.                 'barang' => $this->produkRepository->showEtalase(),
45.                 'barang_paket' => $this->produkRepository->showByJenis('Paket'),
46.                 'barang_timbang' => $this->produkRepository->showByJenis('Timbang'),
47.                 'barang_kemas' => $this->produkRepository->showByJenis('Kemas'),
48.                 'kategori' => $this->kategoriRepository->array(),
49.                 'baseKategori' => $this->baseKategoriRepository->all(),
50.                 'user' => Auth::user()
51.             ];
52.
53.             return view('reseller.profil')
54.                 ->with(compact('data'))
55.                 ->with(compact('kategori'));
56.         }
57.
58.         public function editProfile(){
59.             $data=[
60.                 'user'=>Auth::user()
61.             ];
62.             return view('reseller.ubah-profil')->with(compact('data'));
63.         }
64.
65.         public function _editProfile(Request $request){
```

```
66.             try
67.             {
68.                 DB::beginTransaction();
69.
70.                 $this->userRepository->update($request-
    >all());
71.
72.                 DB::commit();
73.
74.                 return redirect('Reseller/Profil/');
75.             }
76.             catch (\Throwable $th)
77.             {
78.                 DB::rollback();
79.
80.                 return $th;
81.             }
82.         }
83.     }
```

Kode Sumber 5.64 ProfilController

5.3.4.5. TransaksiController

Lapisan ini bertugas untuk mengatur Transaksi Reseller. Kode Sumber 5.65 berikut merupakan implementasi dari lapisan kontrol TransaksiController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\reseller;
4.
5.  use Auth;
6.  use Carbon\Carbon;
7.  use App\Http\Controllers\Controller;
8.  use Illuminate\Http\Request;
9.  use App\Repositories\TransaksiRepository;
```

```
10.  use App\Repositories\DetailTransaksiRepository;
11.  use App\Repositories\KeranjangRepository;
12.  use App\Repositories\DetailKeranjangRepository;
13.  use App\Repositories\AlamatRepository;
14.  use App\Repositories\ProdukRepository;
15.  use App\Repositories\IsiPaketRepository;
16.  use App\Repositories\KeranjangResellerRepository;
17.  use App\Repositories\ParentKeranjangResellerReposit
     ory;
18.  use App\Repositories\UserRepository;
19.  use Uuid;
20.  use DB;
21.
22.  class TransaksiController extends Controller
23.  {
24.      protected $barang;
25.      protected $transaksi;
26.      protected $keranjang;
27.      protected $detailTransaksi;
28.      protected $detailKeranjang;
29.      protected $alamat;
30.      protected $isiPaket;
31.      protected $keranjangReseller;
32.      protected $parentKeranjangReseller;
33.      protected $user;
34.
35.      public function __construct(ProdukRepository $b
     arang, TransaksiRepository $transaksi, KeranjangRep
     ository $keranjang, DetailTransaksiRepository $deta
     ilTransaksi, DetailKeranjangRepository $detailKeran
     jang, AlamatRepository $alamat, IsiPaketRepository
     $isiPaket, KeranjangResellerRepository $keranjangRe
     seller, ParentKeranjangResellerRepository $parentKe
     ranjangReseller, UserRepository $user)
36.      {
37.          $this->middleware('auth');
38.
```

```
39.             $this->transaksi        = $transaksi;
40.             $this-
    >detailTransaksi  = $detailTransaksi;
41.             $this->keranjang        = $keranjang;
42.             $this-
    >detailKeranjang  = $detailKeranjang;
43.             $this->alamat           = $alamat;
44.             $this->barang           = $barang;
45.             $this->isiPaket         = $isiPaket;
46.             $this-
    >keranjangReseller = $keranjangReseller;
47.             $this-
    >parentKeranjangReseller = $parentKeranjangReseller
    ;
48.             $this->user             = $user;
49.         }
50.
51.         public function viewCheckout(Request $request)

52.         {
53.             $alamat=$this->alamat-
    >getAllAlamatByUser(Auth::user()->id);
54.             if(count($alamat)==0){
55.                 return redirect('/Reseller/TambahAlamat
    /');
56.             }
57.             $getBarangs           = $this-
    >keranjangReseller->getTotalById($request-
    >tanggal);
58.             // dd($getBarang);
59.             $totalHarga             = 0;
60.             $detailPaketKeranjang   = array();
61.
62.             foreach($getBarangs as $detailKeranjang)
63.             {
64.                 $getBarang                      = $this
    ->barang->findById($detailKeranjang->id_barang);
```

```
65.
66.                 if($getBarang->jenis == 'Kemas')
67.                 {
68.                     $detailKeranjang->nama          = $detailKeranjang->bobot_kemasan.' '.'Gram'.' '.$getBarang->nama;
69.                     $detailKeranjang->jenis         = $getBarang->jenis;
70.                     $detailKeranjang->satuan        = $getBarang->satuan;
71.                     $detailKeranjang->bobot         = $getBarang->bobot;
72.                     $detailKeranjang->harga_satuan  = $getBarang->harga_jual;
73.                     $detailKeranjang->volume        = $detailKeranjang->volume * ($getBarang->bobot/10);
74.                     $detailKeranjang->diskon        = $getBarang->diskon;
75.                     $detailKeranjang->jenis_diskon  = $getBarang->jenis_diskon;
76.
77.                     $totalHarga = $totalHarga + $detailKeranjang->harga_diskon;
78.                 }
79.                 else if($getBarang->jenis == 'Paket')
80.                 {
81.                     $detailKeranjang->nama          = $getBarang->nama;
82.                     $detailKeranjang->jenis         = $getBarang->jenis;
83.                     $detailKeranjang->satuan        = $getBarang->satuan;
84.                     $detailKeranjang->bobot         = $getBarang->bobot;
85.                     $detailKeranjang->harga_satuan  = $getBarang->harga_jual;
```

```
86.                     $detailKeranjang->isPaket       = $getBarang->is_paket;
87.                     $detailKeranjang->diskon        = $getBarang->diskon;
88.                     $detailKeranjang->jenis_diskon  = $getBarang->jenis_diskon;
89.
90.                     $getIsiPaket = $this->isiPaket->findByIdBarang($detailKeranjang->id_barang);
91.                     $getIsiPaket = array($getIsiPaket);
92.
93.                     for($a=0;$a<sizeof($getIsiPaket[0]);$a++)
94.                     {
95.                         $getBarang = $this->barang->findById($getIsiPaket[0][$a]->id_barang);
96.
97.                         $detailPaketKeranjang[$detailKeranjang->id_barang][$a]['nama']   = $getBarang->nama;
98.                         $detailPaketKeranjang[$detailKeranjang->id_barang][$a]['volume'] = $getIsiPaket[0][$a]->volume;
99.                         $detailPaketKeranjang[$detailKeranjang->id_barang][$a]['satuan'] = $getBarang->satuan;
100.                    }
101.
102.                    $totalHarga = $totalHarga + $detailKeranjang->harga_diskon;
103.                }
104.                else if($getBarang->jenis == 'Timbang')
105.                {
```

```
106.                     $detailKeranjang->nama           = $getBarang->nama;
107.                     $detailKeranjang->jenis          = $getBarang->jenis;
108.                     $detailKeranjang->satuan         = $getBarang->satuan;
109.                     $detailKeranjang->bobot          = $getBarang->bobot;
110.                     $detailKeranjang->harga_satuan   = $getBarang->harga_jual;
111.                     $detailKeranjang->diskon         = $getBarang->diskon;
112.                     $detailKeranjang->jenis_diskon   = $getBarang->jenis_diskon;
113.
114.                     $totalHarga = $totalHarga + $detailKeranjang->harga_diskon;
115.                 }
116.             // dd($totalHargaKemas);
117.
118.             }
119.             // $totalHarga = $totalHarga + 10000;
120.             // dd($this->transaksi->checkTransaksi($request->tanggal));
121.             if($this->transaksi->checkTransaksi($request->tanggal) == null)
122.             {
123.                 $data = [
124.                     'tanggal_pre_order'     => $request->tanggal,
125.                     'isCheckout'            => 0,
126.                     'total_bayar'           => $totalHarga
127.                 ];
128.
129.                 $this->transaksi->halamanCheckoutReseller($data);
```

```
130.           }
131.
132.           else
133.           {
134.               $data = [
135.                   'tanggal_pre_order'     => $request
      ->tanggal,
136.                   'total_bayar'           => $totalHa
      rga
137.               ];
138.
139.               $this->transaksi-
      >updateHalamanCheckout($data);
140.           }
141.
142.           $getAlamat = $this->alamat-
      >getAllAlamatByUser(Auth::user()->id);
143.
144.           $data = [
145.               'keranjang'             => $this-
      >parentKeranjangReseller-
      >getKeteranganKeranjang($request->tanggal),
146.               'detail_keranjang'      => $getBarangs
      ,
147.               'total'                 => $totalHarga
      ,
148.               'detail_paket_keranjang' => $detailPake
      tKeranjang,
149.               'alamat'                => $getAlamat

150.           ];
151.
152.           // dd($request->tanggal);
153.
154.           return view('reseller.checkout')-
      >with(compact('data'));
155.       }
```

```
156.
157.       public function submitCheckout(Request $request
       )
158.       {
159.           // dd($request->alamat);
160.           // $getKeranjang           = $this-
       >keranjang->getIDKeranjang();
161.
162.           // $getDetailKeranjang     = $this-
       >detailKeranjang-
       >getDetailKeranjang($getKeranjang);
163.           $getBarangs           = $this-
       >keranjangReseller->getTotalById($request-
       >tanggal);
164.           $getTransaksi           = $this->transaksi-
       >checkTransaksi($request->tanggal);
165.           $nomor=$getTransaksi->nomor_invoice;
166.
167.           foreach($getBarangs as $detail)
168.           {
169.               // dd($detail->bobot_kemasan);
170.               $data = [
171.                   'id_barang'     => $detail-
       >id_barang,
172.                   'harga'         => $detail-
       >harga,
173.                   'harga_diskon'  => $detail-
       >harga_diskon,
174.                   'volume'        => $detail-
       >volume,
175.                   'id_transaksi'  => $getTransaksi-
       >id,
176.                   'bobot_kemasan' => $detail-
       >bobot_kemasan
177.               ];
178.
```

```
179.                 $this->detailTransaksi->inputDetailTransaksi($data);
180.             }
181.
182.
183.             $data = [
184.                 'keterangan' => $request->keterangan,
185.                 'id_alamat'  => $request->alamat
186.             ];
187.
188.             $this->transaksi->purchaseCheckout($data, $request->tanggal);
189.
190.             // $this->keranjang->hapusKeranjang($getKeranjang);
191.             // $this->detailKeranjang->hapusDetailKeranjangByIdKeranjang($getKeranjang);

192.             $this->parentKeranjangReseller->settingFlag(1, $getTransaksi->id, $request->tanggal);
193.
194.             return view('reseller.transaksi-sukses')->with(compact('nomor'));
195.         }
196.
197.         public function showTransaksi($tipe){
198.
199.             $alltype=[
200.                 [
201.                     'status'=>"BelumDibayar",
202.                     'status2' => "Belum Dibayar"
203.                 ],
204.                 [
205.                     'status'=>"Proses",
206.                     'status2' => "Dalam Proses"
207.                 ],
```

```
208.                [
209.                    'status'=>"Selesai",
210.                    'status2' => "Selesai"
211.                ],
212.                [
213.                    'status'=>"Dibatalkan",
214.                    'status2' => "Dibatalkan"
215.                ],
216.            ];
217.            if($tipe=="BelumDibayar"){
218.                $tipe2="Belum Dibayar";
219.                $transaksi = $this->transaksi-
    >getAllTransaksiByUserAndNotPay(Auth::user()-
    >id);
220.            }
221.            else if($tipe=="Proses"){
222.                $tipe2="Dalam Proses";
223.                $transaksi = $this->transaksi-
    >getAllTransaksiByUserAndInProcess(Auth::user()-
    >id);
224.            }
225.            else if($tipe=="Selesai"){
226.                $tipe2="Selesai";
227.                $transaksi = $this->transaksi-
    >getAllTransaksiByUserAndIsFinish(Auth::user()-
    >id);
228.            }
229.            else if($tipe=="Dibatalkan"){
230.                $tipe2="Dibatalkan";
231.                $transaksi = $this->transaksi-
    >getAllTransaksiByUserAndIsCancelled(Auth::user()-
    >id);
232.            }
233.            else{
234.                return redirect('/');
235.            }
236.            $data = [
```

```
237.                 'alltype'=>$alltype,
238.                 'tipe'=>$tipe,
239.                 'tipe2'=>$tipe2,
240.                 'transaksi'=>$transaksi,
241.                 'rekening'=>$this->user->getVeggo()
242.             ];
243.
244.             return view('reseller.transaksi')-
    >with('data',$data);
245.         }
246.
247.         public function _filterTanggal(Request $request
    ){
248.             session(['tanggal_pre_order' => $request-
    >input('tanggal')]);
249.             return redirect('/reseller/Transaksi');
250.         }
251.
252.         public function kirimBukti($id_transaksi,Reques
    t $request)
253.         {
254.
255.             // dd('masuk bang');
256.             $file = $request->file('foto_bukti');
257.             // dd($file);
258.             $filename = Uuid::generate(4)-
    >string.'.'.$file->getClientOriginalExtension();
259.             $path = public_path().'/img/foto_bukti';
260.             $file->move($path,$filename);
261.
262.             $this->transaksi-
    >kirimBukti($id_transaksi, $filename);
263.
264.             return redirect()->back();
265.
266.         }
267.
```

```
268.      public function detailTransaksi($id_transaksi){
269.          //get id parent dari id transaksi
270.          $parent=$this->parentKeranjangReseller-
      >getFromIdTransaksi($id_transaksi);
271.          // dd($parent);
272.          //get barang dari id parent
273.          $detail =[
274.              'transaksi'=>$parent
275.          ];
276.          // dd($detail);
277.          return view('reseller.detail-transaksi')
278.              ->with(compact('detail'));
279.      }
280.
281.      public function konfirmasiTransaksi($id_transak
      si){
282.          $this->transaksi-
      >konfirmasiTransaksi($id_transaksi);
283.          // dd($detail);
284.          return redirect()->back();
285.      }
286.  }
```

Kode Sumber 5.65 TransaksiController

5.3.4.6. UsersController

Lapisan ini bertugas untuk mengatur Pembeli Reseller. Kode Sumber 5.66 berikut merupakan implementasi dari lapisan kontrol UsersController:

```
1.  <?php
2.
3.  namespace App\Http\Controllers\Reseller;
4.
5.  use Carbon\Carbon;
```

```
6.  use App\Http\Controllers\Controller;
7.  use Illuminate\Http\Request;
8.  use App\Repositories\KeranjangResellerRepository;
9.  use App\Repositories\ParentKeranjangResellerReposit
    ory;
10. use App\Repositories\TanggalRepository;
11. use Illuminate\Support\Facades\DB;
12.
13. use Illuminate\Support\Facades\Auth;
14.
15. class UsersController extends Controller
16. {
17.     private $keranjangReseller;
18.
19.     public function __construct(KeranjangResellerRe
    pository $keranjangReseller, ParentKeranjangReselle
    rRepository $parentKeranjangReseller, TanggalReposi
    tory $tanggal)
20.     {
21.         $this->middleware('auth');
22.         $this->middleware('reseller');
23.
24.         $this-
    >keranjangReseller = $keranjangReseller;
25.         $this-
    >parentKeranjangReseller = $parentKeranjangReseller
    ;
26.         $this->tanggal = $tanggal;
27.     }
28.
29.     public function show($id, $date)
30.     {
31.         $data = [
32.             'tanggal' =>$this-
    >getTanggalPengiriman(),
33.             'tanggals'=>$date,
```

```
34.                 'keranjangReseller' => $this-
    >parentKeranjangReseller-
    >getAllUserByUserandStatus(Auth::user()-
    >id, $date)
35.             ];
36.
37.             return view('reseller.users')-
    >with(compact('data'));
38.         }
39.         public function getTanggalPengiriman()
40.         {
41.
42.             $getListHari=$this->tanggal-
    >get_tanggal();
43.
44.             for($a=0;$a<sizeof($getListHari);$a++)
45.             {
46.                 $tgl=$getListHari[$a]['tanggal'];
47.                 // string
48.                 $finalListHari[$a]['tanggal'] = Carbon:
    :parse($tgl)->format('D, d F Y');
49.                 //tanggal value
50.                 $finalListHari[$a]['tanggal_value'] = C
    arbon::parse($tgl)->format('Y-m-d');
51.
52.                 // dd($finalListHari[$a]['tanggal_value
    ']);
53.             }
54.
55.
56.
57.             foreach ($finalListHari as $key => $value)

58.             {
59.                 $finalListHari[$key] = (object) $value;

60.             }
```

```
61.
62.             // dd($finalListHari);
63.
64.             return $finalListHari;
65.         }
66.
67.         public function hapus($id)
68.         {
69.             $this->keranjangReseller->hapusUser($id);
70.             $this->parentKeranjangReseller-
    >hapusUser($id);
71.
72.             return redirect()->back();
73.         }
74.
75.         public function detail($id)
76.         {
77.             $keranjang=$this->keranjangReseller-
    >getUserById($id);
78.             $total=0;
79.             foreach($keranjang as $ker){
80.                 $total+=$ker->harga_diskon;
81.             }
82.             $data=[
83.                 'keranjangReseller'=>$keranjang,
84.                 'dataPembeli'=>$this-
    >parentKeranjangReseller->getUserById($id),
85.                 'total'=>$total,
86.
87.             ];
88.             // dd($data['keranjangReseller']);
89.             return view('reseller.detail-user')-
    >with(compact('data'));
90.         }
91.
92.         public function hapusBarang($id_barang, $id_par
    ent){
```

```
93.             $this->keranjangReseller-
      >hapusBarang($id_barang, $id_parent);
94.             return redirect()->back();
95.         }
96.     }
```

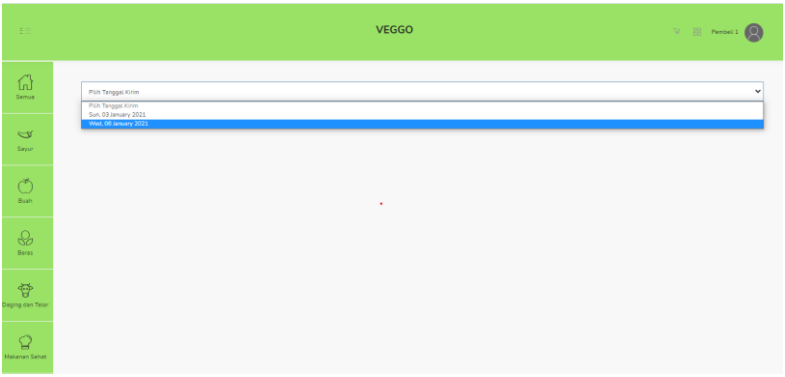
Kode Sumber 5.66 UsersController

5.4. Implementasi Antarmuka Pengguna

Pada bagian ini kami akan menampilkan antarmuka halaman yang ada pada aplikasi VEGGO. Berikut tampilan antarmuka aplikasi VEGGO :

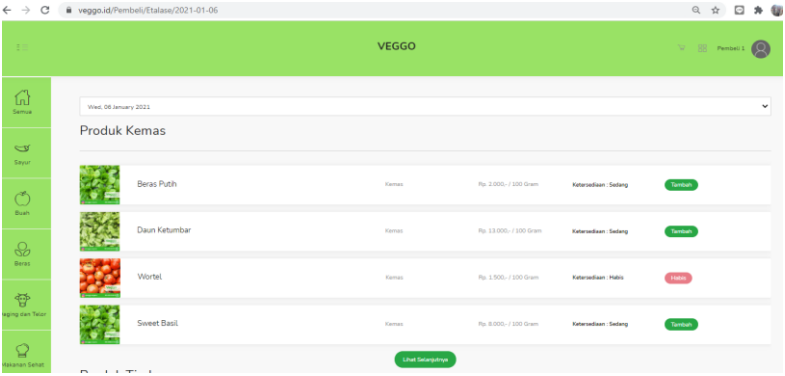
5.4.1. Antarmuka Pembeli

5.4.1.1. Antarmuka Pemilihan Tanggal Pengiriman



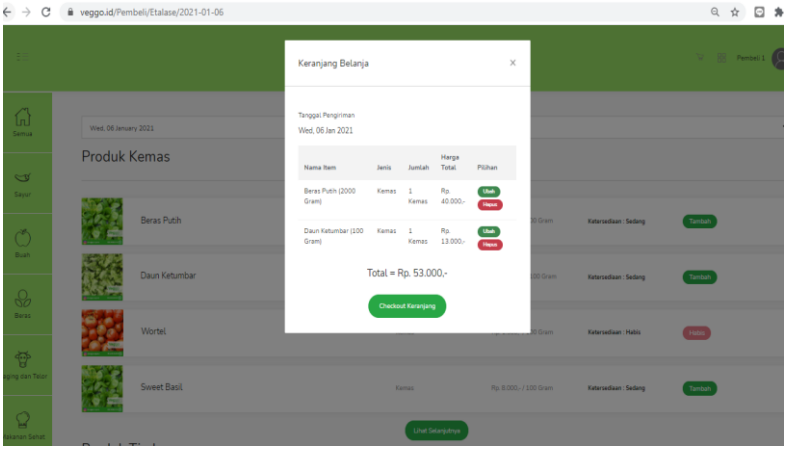
Gambar 5.1 Antarmuka Pemilihan Tanggal Pengiriman

5.4.1.2. Antarmuka Etalase Barang



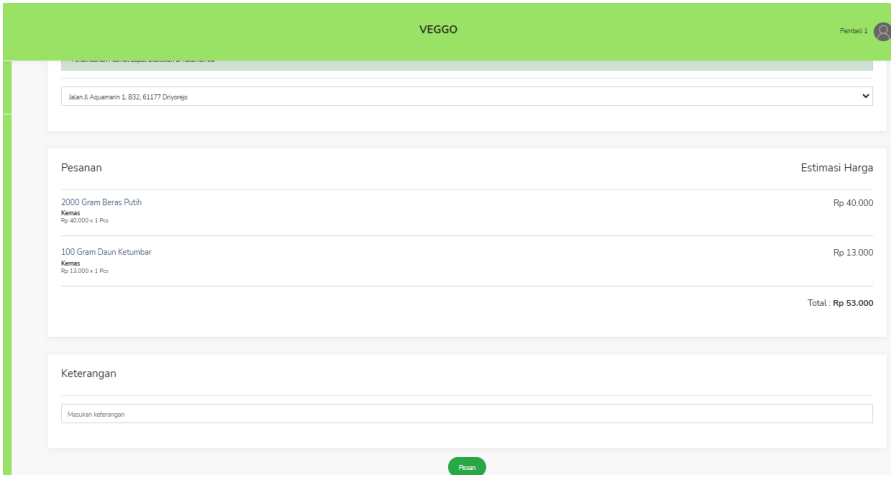
Gambar 5.2 Antarmuka Etalase Barang

5.4.1.3. Antarmuka *Checkout* Keranjang



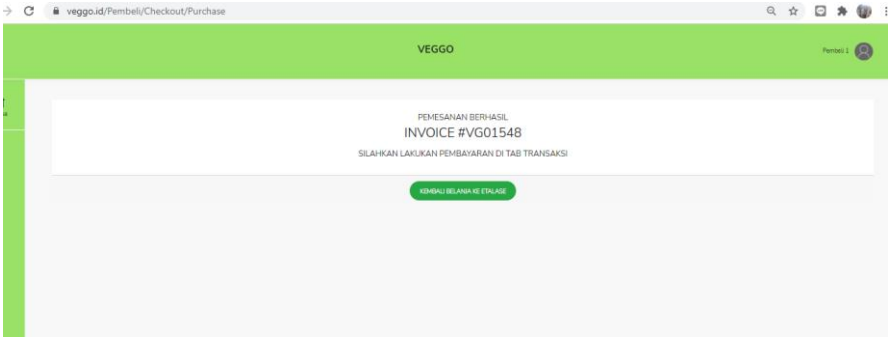
Gambar 5.3 Antarmuka Checkout Keranjang

5.4.1.4. Antarmuka Konfirmasi Pembelian



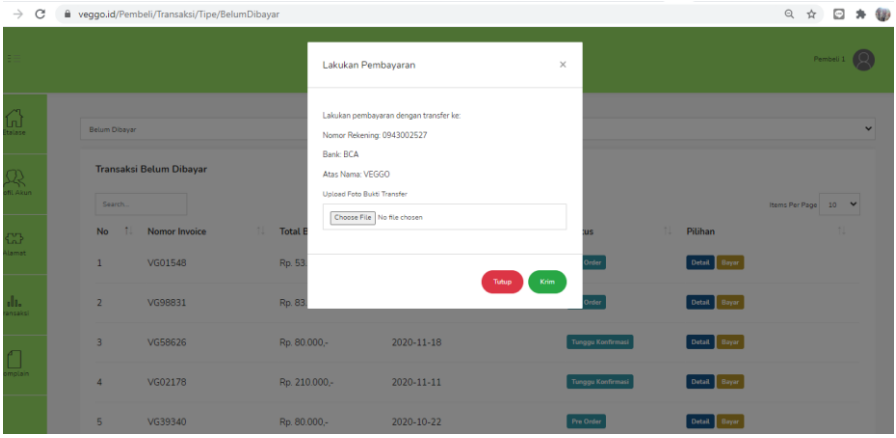
Gambar 5.4 Antarmuka Konfirmasi Pembelian

5.4.1.5. Antarmuka Kode Transaksi



Gambar 5.5 Antarmuka Kode Transaksi

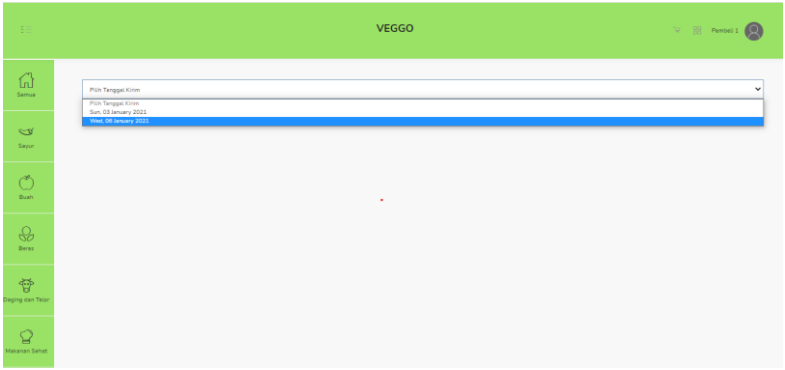
5.4.1.6. Antarmuka *Upload* Pembayaran



Gambar 5.6 Antarmuka Upload Pembayaran

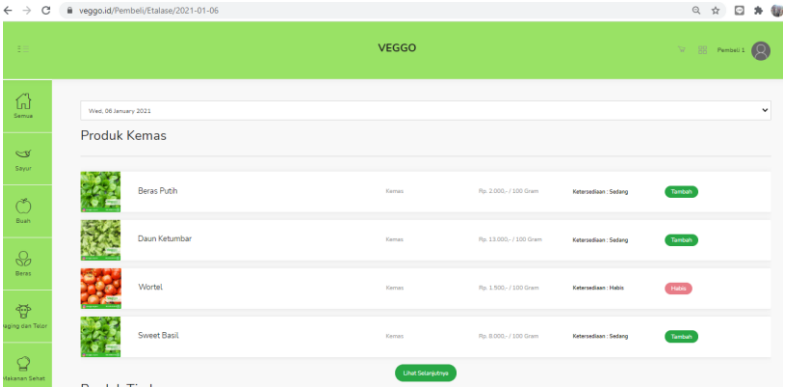
5.4.2. Antarmuka Reseller

5.4.2.1. Antarmuka Pemilihan Tanggal Pengiriman



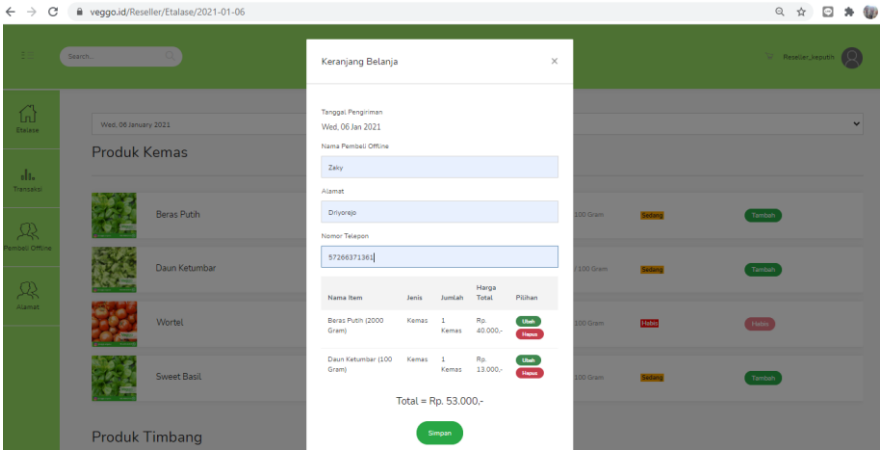
Gambar 5.7 Antarmuka Pemilihan Tanggal Pengiriman

5.4.2.2. Antarmuka Etalase Barang



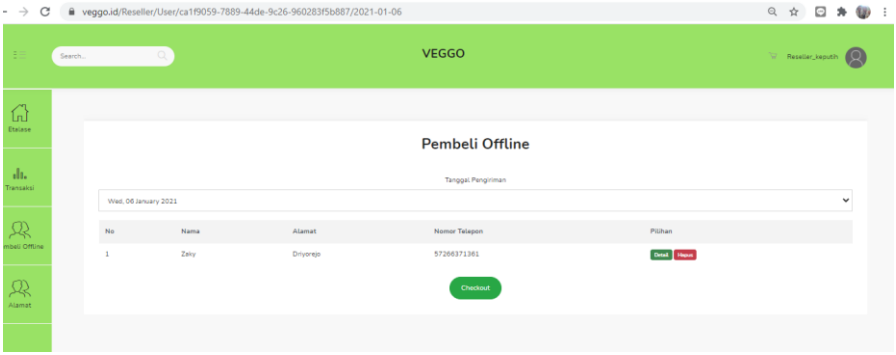
Gambar 5.8 Antarmuka Etalase Barang

5.4.2.3. Antarmuka Pengisian Data Pembeli *Offline*



Gambar 5.9 Antarmuka Pengisian Data Pembeli Offline

5.4.2.4. Antarmuka *Checkout* Pembelian *Offline*

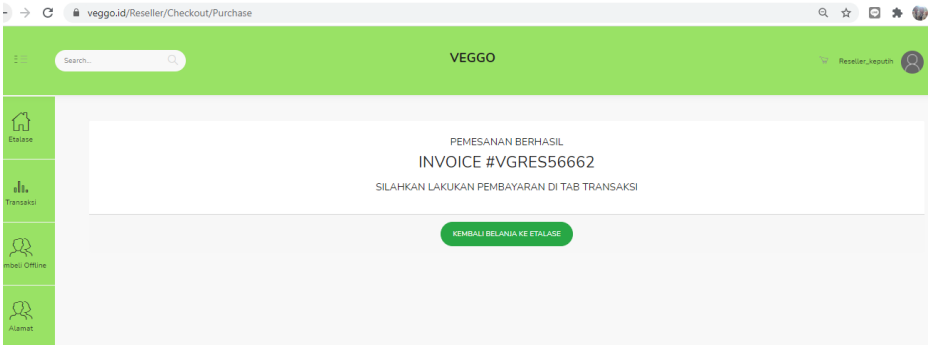


Gambar 5.10 Antarmuka Checkout Pembelian Offline

5.4.2.5. Antarmuka Konfirmasi Pembelian

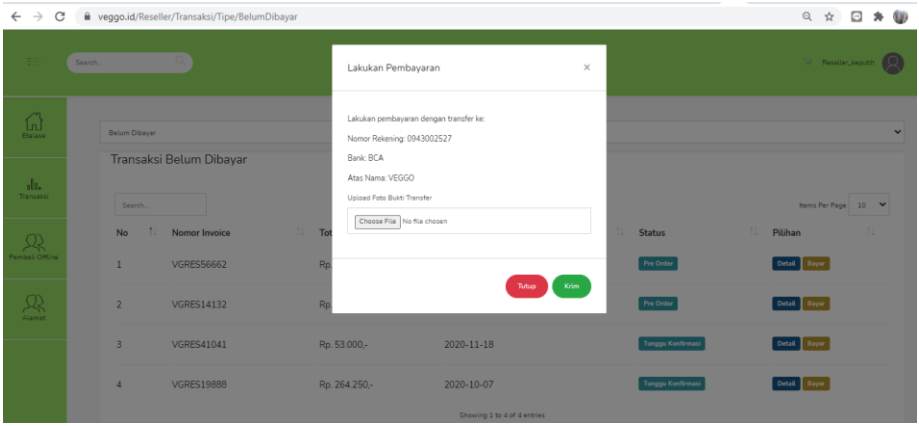
Gambar 5.11 Antarmuka Konfirmasi Pembelian

5.4.2.6. Antarmuka Kode Transaksi



Gambar 5.12 Antarmuka Kode Transaksi

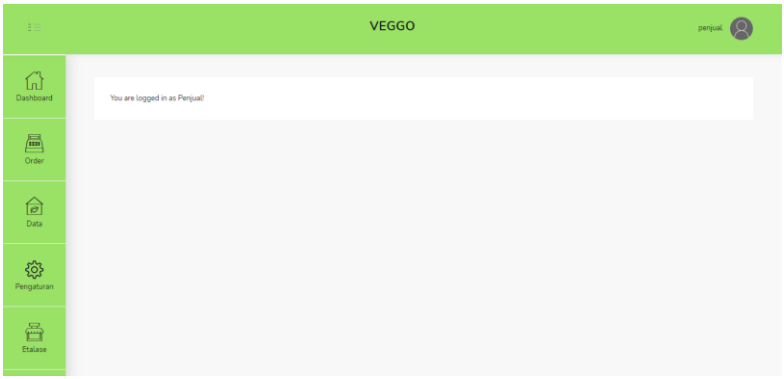
5.4.2.7. Antarmuka *Upload* Pembayaran



Gambar 5.13 Antarmuka Upload Pembayaran

5.4.3. Antarmuka Penjual

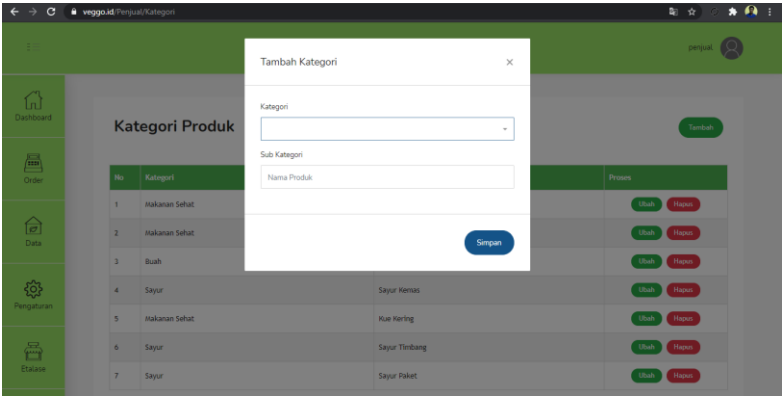
5.4.3.1. Antarmuka *Home* Penjual



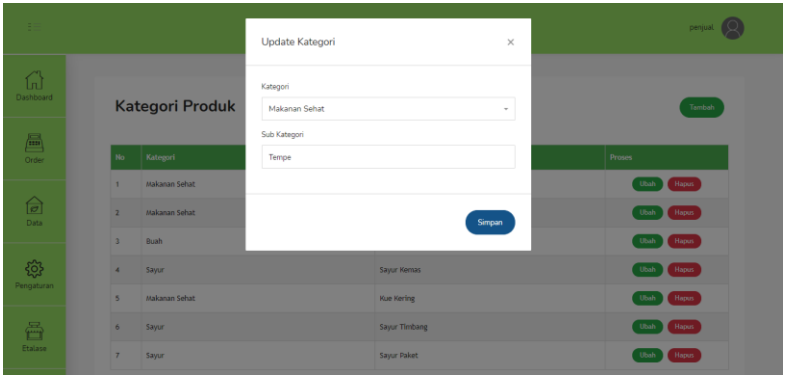
Gambar 5.14 Antarmuka Home Penjual

5.4.3.2. Antarmuka Kelola Kategori

Gambar 5.15 Antarmuka Kelola Kategori

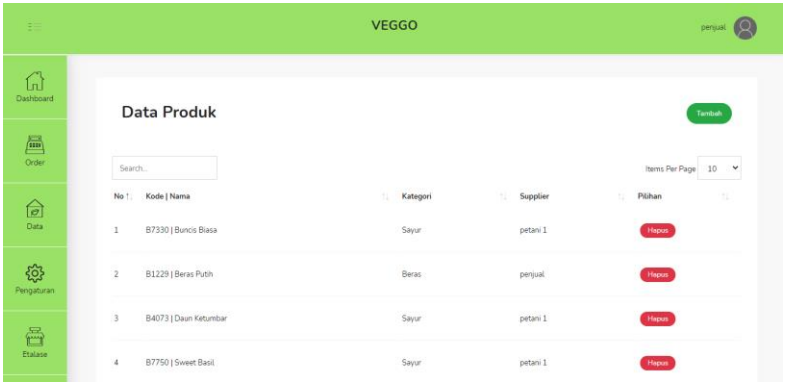


Gambar 5.16 Antarmuka Tambah Kategori

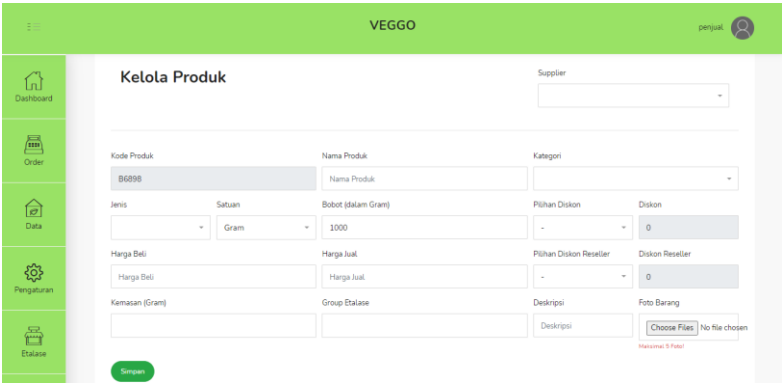


Gambar 5.17 Antarmuka Ubah Kategori

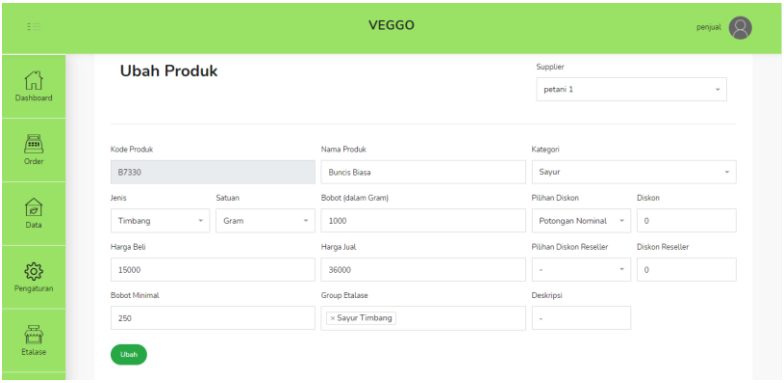
5.4.3.3. Antarmuka Kelola Produk



Gambar 5.18 Antarmuka Kelola Produk

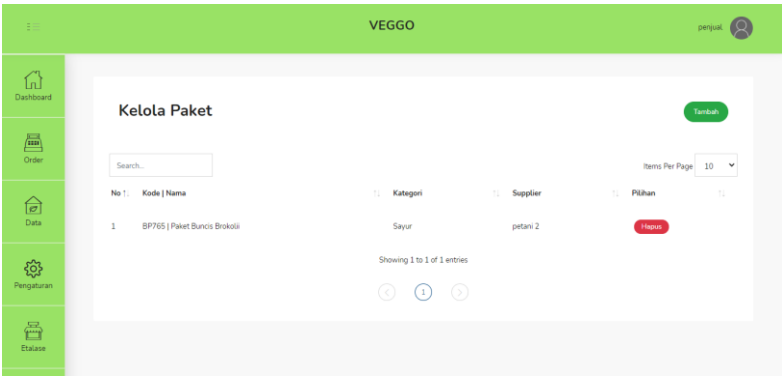


Gambar 5.19 Antarmuka Tambah produk

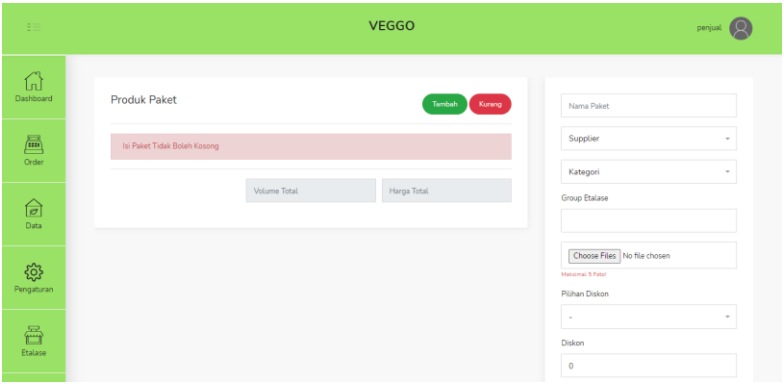


Gambar 5.20 Antarmuka Ubah Produk

5.4.3.4. Antarmuka Kelola Paket



Gambar 5.21 Antarmuka Kelola Paket



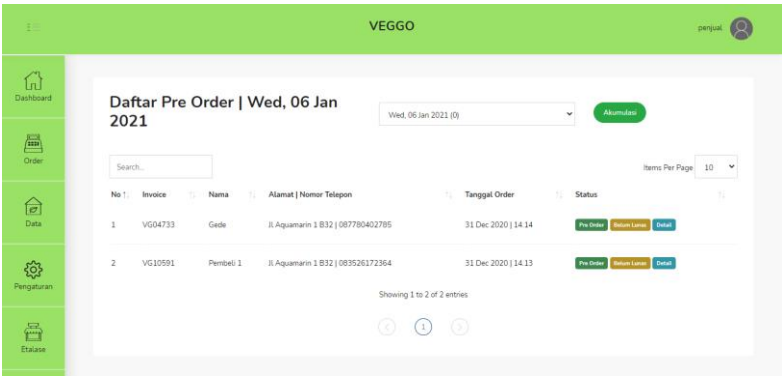
Gambar 5.22 Antarmuka Tambah Paket

Gambar 5.23 Antarmuka Ubah paket

5.4.3.5. Antarmuka Kelola Tanggal Pengiriman

Gambar 5.24 Antarmuka Kelola Tanggal pengiriman

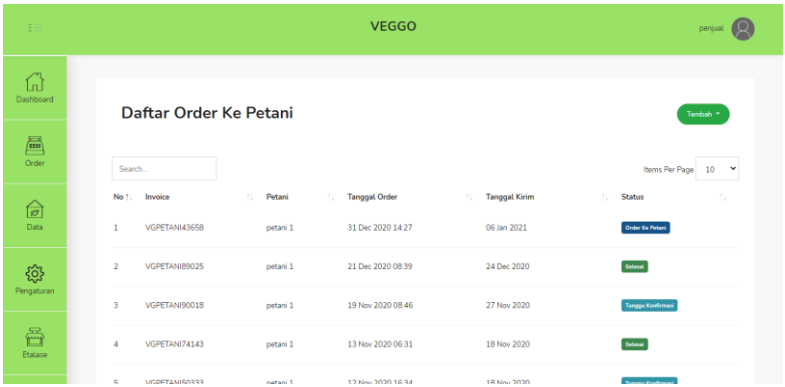
5.4.3.6. Antarmuka *Pre Order* Pembeli



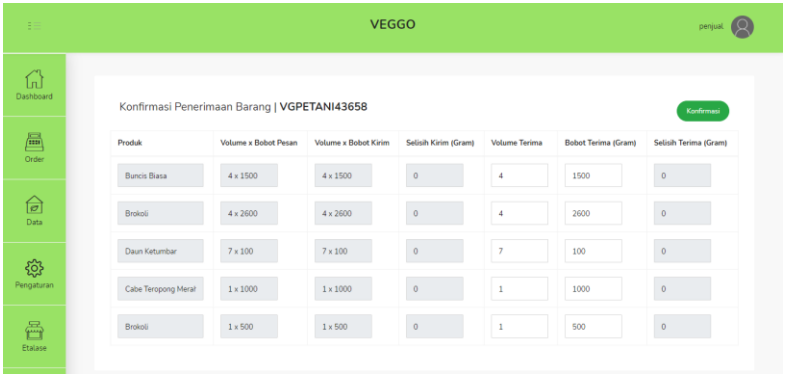
Gambar 5.25 Antarmuka Pre Order Pembeli

Gambar 5.26 Antarmuka Akumulasi Pre Order Pembeli

5.4.3.7. Antarmuka *Order* ke Petani

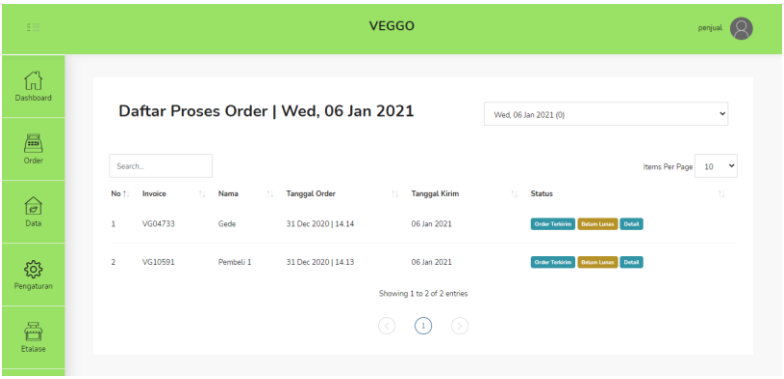


Gambar 5.27 Antarmuka Order ke Petani

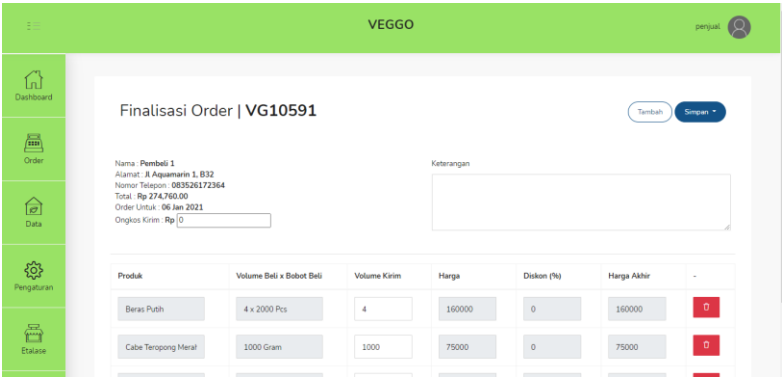


Gambar 5.28 Antarmuka Konfirmasi Order ke Petani

5.4.3.8. Antarmuka Proses *Order* ke Pembeli

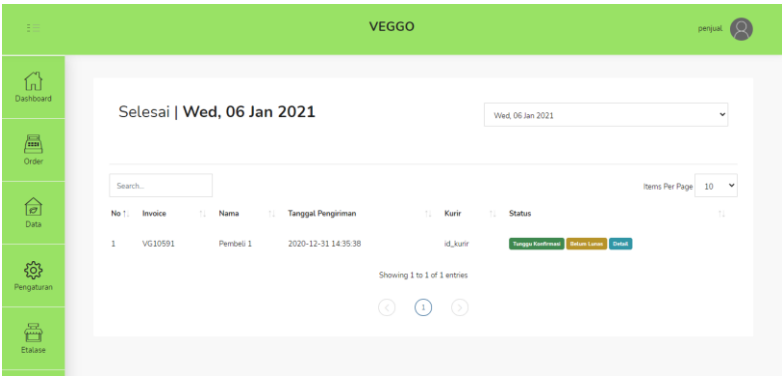


Gambar 5.29 Antarmuka Proses Order ke Pembeli



Gambar 5.30 Antarmuka Finalisasi Order ke Pembeli

5.4.3.9. Antarmuka *Order* Selesai



Gambar 5.31 Antarmuka Order Selesai

5.4.3.10. Antarmuka Tagihan Petani

Gambar 5.32 Antarmuka Tagihan Petani

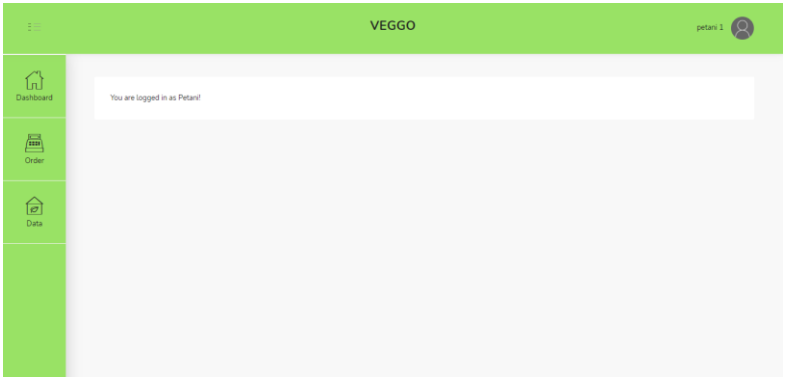
5.4.3.11. Antarmuka Rekap Transaksi

Gambar 5.33 Antarmuka Rekap Transaksi Harian

Gambar 5.34 Antarmuka Rekap Transaksi Bulanan

5.4.4. Antarmuka Petani

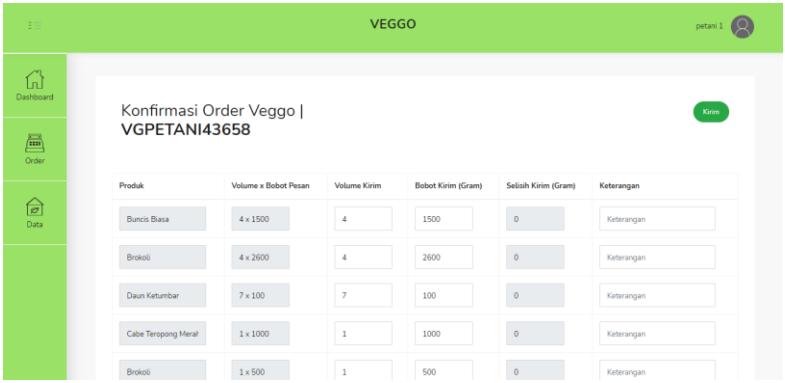
5.4.4.1. Antarmuka *Home* Petani



Gambar 5.35 Antarmuka Home Petani

5.4.4.2. Antarmuka Penerimaan *Order* ke Petani

Gambar 5.36 Antarmuka Penerimaan Order ke Petani

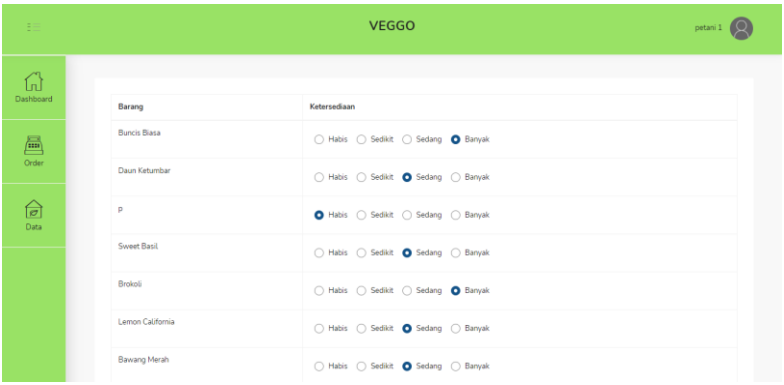


Gambar 5.37 Antarmuka Konfirmasi Penerimaan Order ke Petani

5.4.4.3. Antarmuka Tagihan ke Penjual

Gambar 5.38 Antarmuka Tagihan ke Penjual

5.4.4.4. Antarmuka Kelola Ketersediaan



Gambar 5.38 Antarmuka Kelola Ketersediaan

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba terhadap Aplikasi VEGGO. Pengujian dilakukan untuk memastikan kualitas perangkat lunak yang dibangun dan kesesuaian hasil eksekusi perangkat lunak dengan analisis dan perancangan perangkat lunak.

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap Aplikasi Sistem Marketplace VEGGO guna mengetahui beberapa hal berikut ini:

a. Menguji kesesuaian dan ketepatan fungsionalitas dari seluruh sistem aplikasi.
b. Menguji kesesuaian fitur yang dikembangkan dalam aplikasi VEGGO.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memperhatikan beberapa hasil yang diharapkan berikut:

a. Kemampuan aplikasi untuk melakukan pembelian barang
b. Kemampuan aplikasi untuk melakukan order ke petani
c. Kemampuan aplikasi untuk mengatur barang dagangan sesuai tanggal pengiriman.
d. Kemampuan aplikasi untuk melakukan rekap penjualan
e. Kemampuan aplikasi untuk menyimpan transaksi pembelian
f. Kemampuan aplikasi untuk mengkonfirmasi kuantitas atau banyaknya pesanan yang bisa dikirim ke Penjual

g. Kemampuan aplikasi untuk mengirimkan bukti pembayaran ke Penjual
h. Kemampuan aplikasi untuk mengkonfirmasi barang yang telah sampai
i. Kemampuan aplikasi memenuhi kebutuhan fungsionalitas lainnya seperti memasukkan data, mengubah data, menghapus data, dan melihat detail data.
j. Kesesuaian dalam memenuhi kebutuhan non-fungsionalitas aplikasi, yaitu:
    - Dapat diakses melalui internet.
    - Sistem memiliki tampilan (antarmuka) yang mudah dipahami.

6.3. Skenario Pengujian

Skenario pengujian dilakukan dengan melakukan peran sebagai agen maupun admin yang akan menjalankan fitur-fitur dan seluruh kebutuhan fungsional dari sistem. Langkah-langkah untuk setiap kebutuhan fungsionalitas yaitu sebagai berikut:

A. Sebagai Pembeli:
    1. Pembeli membuka aplikasi VEGGO.
    2. Pembeli melakukan *login* menuju ke halaman utama untuk memilih tanggal pengiriman.
    3. Pembeli memilih barang di etalase untuk ditambahakan ke keranjang.
    4. Pembeli masuk ke halaman *checkout* keranjang dan mengecek barang yang telah ditambahkan.
    5. Pembeli masuk ke halaman konfirmasi pembelian untuk memastikan data pembelian.
    6. Pembeli menekan tombol Pesan.
    7. Pembeli mendapatkan kode transaksi.

8. Pembeli melakukan pembayaran dengan cara mengupload bukti pembayaran.
9. Pembeli melakukan konfirmasi barang telah sampai.

B. Sebagai Reseller:
    1. Reseller membuka aplikasi VEGGO.
    2. Reseller melakukan *login* menuju ke halaman utama untuk memilih tanggal pengiriman.
    3. Reseller memilih barang di etalase untuk ditambahakan ke keranjang.
    4. Reseller masuk ke halaman *checkout* keranjang dan mengecek barang yang telah ditambahkan.
    5. Reseller masuk ke halaman konfirmasi pembelian untuk memastikan data pembelian.
    6. Reseller menekan tombol Pesan.
    7. Reseller mendapatkan kode transaksi.
    8. Reseller melakukan pembayaran dengan cara mengupload bukti pembayaran.
    9. Reseller melakukan konfirmasi barang telah sampai.

C. Sebagai Penjual:
    1. Penjual membuka aplikasi VEGGO.
    2. Penjual menambah, mengubah, atau menghapus kategori.
    3. Penjual menambah, mengubah atau menghapus Produk.
    4. Penjual menambah, mengubah, atau menghapus Paket.
    5. Penjual membuka tanggal pengiriman.
    6. Penjual memilih Produk/Paket di tanggal pengiriman yang sudah buka.

7. Penjual melihat *Pre-Order* Pembeli.
8. Penjual melakukan akumulasi *Pre-Order* Pembeli.
9. Penjual mengirimkan *order* ke Petani.
10. Penjual mengkonfirmasi *order* ke Petani.
11. Penjual melakukan klaim *order* ke Petani.
12. Penjual melihat *order* Pembeli.
13. Penjual melakukan finalisasi o*rder* ke Pembeli.
14. Penjual melihat *order* Pembeli yang selesai.
15. Penjual melihat rekap tagihan ke Petani
16. Penjual melihat rekap barang.
17. Penjual *logout* dari aplikasi VEGGO.

D. Sebagai Petani:
    1. Petani membuka aplikasi VEGGO.
    2. Petani melihat *order* dari Penjual.
    3. Petani mengisi dan mengirimkan *order* ke Penjual.
    4. Petani melihat klaim.
    5. Petani mengatur stok produk yang dimilikinya.
    6. Petani *logout* dari aplikasi VEGGO.

6.4. Evaluasi Pengujian

Hasil pengujian dilakukan terhadap pengamatan mengenai perilaku sistem aplikasi VEGGO terhadap kasus skenario uji coba. Pengujian dilakukan oleh pihak pengembang, pengguna, VEGGO, dan pembimbing lapangan. Tabel 6.1 di bawah ini menjelaskan hasil uji coba terhadap aplikasi yang telah dibuat.

Tabel 6.1. Hasil Evaluasi Pengujian

Kriteria Pengujian	**Hasil Pengujian**
Dapat melakukan transaksi pembelian	Terpenuhi
Dapat mengupload bukti pembayaran ke penjual	Terpenuhi
Dapat melakukan order ke petani	Terpenuhi
Mengatur tanggal pengiriman yang tersedia beserta produk yang tampil di etalase pembeli	Terpenuhi
Dapat melakukan rekap penjualan	Terpenuhi

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan pengembangan aplikasi pada kegiatan kerja praktek di VEGGO adalah sebagai berikut:

a. Aplikasi yang dibangun telah sesuai dengan permintaan.
b. Dengan adanya aplikasi VEGGO, *Client* dapat dengan mudah memasarkan produknya, mengelola data dan mengelola proses bisnisnya.

7.2. Saran

Saran untuk pengembangan sistem aplikasi VEGGO adalah sebagai berikut:

a. Pada segi Penjual, perlu sebuah fitur untuk menyimpan rekap transaksi dalam bentuk pdf.
b. Perlu diadakan analisis dan diskusi bersama dengan klien untuk memastikan dan mendapatkan kebutuhan serta solusi yang tepat.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] “Instagram Veggo” [Online]. Available: https://www.instagram.com/veggo.organic. [diakses pada tanggal 16 Juli 2020]

[2] "Laravel," [Online]. Available: https://laravel.com/docs/6.x. [diakses pada tanggal 14 Juli 2020].

[3] “JQuery Docs” [Online]. Available: https://dore-jquery-docs.coloredstrategies.com/docs/gettingstarted/introduction [diakses pada tanggal 14 Juli 2020]

[4] “Profil DPTSI” [Online]. Available: https://www.its.ac.id/dptsi/id/tentang-dptsi/. [diakses pada tanggal 12 Januari 2021]

[5] "Apa itu Apache?" [Online]. Available: https://www.hostinger.co.id/tutorial/apa-itu-apache/. [diakses pada tanggal 14 Juli 2020].

[6] "Website VEGGO" [Online]. Available: https://veggo.id/. [diakses pada tanggal 02 November 2020].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS I

Nama	: I Gede Agung Krisna Pamungkas
Tempat, Tanggal Lahir	: Surabaya, 26 Desember 1999
Jenis Kelamin	: Laki-laki
Agama	: Hindu
Status	: Belum Menikah
Alamat Asal	: jl Aquamarin 1 B32 KBD Gresik
Telepon	: +6287780402785
Email	: igedeagung25@gmail.com

PENDIDIKAN FORMAL

2017-sekarang	: Mahasiswa S1 Informatika ITS
2014-2017	: SMA Negeri 9 Surabaya
2011-2014	: SMP Negeri 28 Surabaya
2005-2011	: SDN Balasklumprik II Surabaya

KEMAMPUAN

- *Web Programmin*g (HTML , PHP , Python , CSS , Javascript)
- *Programming* (C , C++ , Python)
- *Database Management* (Oracle, MySQL, SQL Server)
- *Software* Perkantoran (Ms. Office Word , Excel , Powerpoint)

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2017
Semester	: 7 (Tujuh)
IPK	: 3.65 (Semester 6)

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS II

Nama	: Zaky Thariq H
Tempat, Tanggal Lahir	: Sisdoarjo, 04 juni 1999
Jenis Kelamin	: Laki-laki
Agama	: Islam
Status	: Belum Menikah
Alamat Asal	: Sedati Permai GG/22, Sedati, Sidoarjo
Telepon	: +6281231473020
Email	: zakythariq@gmail.com

PENDIDIKAN FORMAL

2017-sekarang	: Mahasiswa S1 Informatika ITS
2014-2017	: SMAN 1 Gedangan
2011-2014	: SMPN 1 Sedati
2005-2011	: SDN Pabean 1

KEMAMPUAN

- *Web Programmin*g (HTML, PHP, CSS)
- *Programming* (C, C++)
- *Database Management* (MySQL, Oracle)
- *Software* Perkantoran (Ms. Office Word , Excel , Powerpoint)
- Bahasa (Indonesia, Inggris)

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2017
Semester	: 7 (Tujuh)
IPK	: 3.30 (Semester 6)

[Halaman ini sengaja dikosongkan]